AD-A266 370

[REPORT DOCUMEN]TATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final Report 01 Jul 88 - 31 Jul 92

4. TITLE AND SUBTITLE
A Joint industry-University program for computational plasma

5. FUNDING NUMBERS
F49620-88-C-0107

6. AUTHOR(S)
Dr Bruce Goplen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Mission Research Corporation
8560 Cinderbed Suite 700
Newington, VA 22122

8. PERFORMING ORGANIZATION REPORT NUMBER
[illegible] 290

DTIC ELECTE JUN 2 9 1993 S A D

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR NE
110 Duncan Avenue Suite B112
Bolling AFB DC 20332-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
2301/A8

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

SEE ATTACH PAGE

This document has been approved for public release and sale; its distribution is unlimited.

93-14757

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

PREFACE

This Manual assumes that the reader may have no prior experience with SOS, but that he is generally acquainted with computational physics methods. The more advanced reader may wish to omit Sections 1, 2, and 3 and focus upon Sections 4, 5, and 6.

Section 1 of the SOS User's Manual discusses the capabilities of SOS. The reader should use this section to determine whether SOS can simulate the physics that he desires. Section 2 of the Manual discusses the complexities of simulating realistic problems, including a discussion of the tradeoffs and approximations that must be made. The reader should use this section to learn about defining and interpreting simulations to maximize their meaningfulness. Section 3 covers issues relating to communicating with SOS, including a step- by-step guide to running the code. The central focus of the chapter is a presentation of the SOS Command Language. Also covered is SOS's response to user errors.

Section 4 is a summary of the conventions used in SOS. Many of the commands use the same concept and same format. Section 5 presents a detailed description of SOS commands in alphabetical order. It describes the impact of each command on the simulation, the meaning of each command argument, and the relationships between commands. Section 6 is an annotated sample simulation, including the input file defining the simulation and the output from SOS. Besides being a good example of the use of SOS, the example can be used as a test of proper installation at the local site.

Appendix A is a summary of the SOS commands. It is intended to be copied and kept handy as a quick reference for checking the format for individual commands and for quickly locating commands that perform desired functions.

A detailed description of the mathematical algorithms is available in the SOS Reference Manual on Algorithms.

(2)

Mission Research Corporation

MRC/WDC-R-283

SOS USER'S MANUAL

A JOINT INDUSTRY-UNIVERSITY PROGRAM FOR COMPUTATIONAL PLASMA

Bruce Goplen
Larry Ludeking
David Smithe
Gary Warren

Technical Report

October 1991

Prepared for: Air Force Office of Scientific Research
Bolling AFB
Washington, D.C. 20332-6448

Contract No. F49620-88-C-0107

Prepared by: MISSION RESEARCH CORPORATION
8560 Cinderbed Road, Suite 700
Newington, Virginia 22122

Research sponsored by the Air Force Office of Scientific Research, under Contract F49620-88-C-0107. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Copy____ MRC/WDC-R-283

SOS USER'S MANUAL

Bruce Goplen
Larry Ludeking
David Smithe
Gary Warren

Technical Report

October 1991

Prepared for: Air Force Office of Scientific Research
Bolling AFB
Washington, D.C. 20332-6448

Contract No. F49620-88-C-0107

Prepared by: MISSION RESEARCH CORPORATION
8560 Cinderbed Road, Suite 700
Newington, Virginia 22122

Copy____ MRC/WDC-R-283

SOS USER'S MANUAL

Bruce Goplen
Larry Ludeking
David Smithe
Gary Warren

Technical Report

October 1991

Prepared for: Air Force Office of Scientific Research
Bolling AFB
Washington, D.C. 20332-6448

Contract No. F49620-88-C-0107

Prepared by: MISSION RESEARCH CORPORATION
8560 Cinderbed Road, Suite 700
Newington, Virginia 22122

REVISIONS

This User's Manual documents SOS, Version - October 1991. The previous version was October 1990. The following new commands have been added:

CALL
ELSE
ENDIF
ENGRID
IF
JOURNAL
MESSAGE
PAUSE
RETURN
SNAPGRID
UNGRID
Z

In addition, there have been changes, both minor and major, to many of the other commands.

PREFACE

This Manual assumes that the reader may have no prior experience with SOS, but that he is generally acquainted with computational physics methods. The more advanced reader may wish to omit Sections 1, 2, and 3 and focus upon Sections 4, 5, and 6.

Section 1 of the SOS User's Manual discusses the capabilities of SOS. The reader should use this section to determine whether SOS can simulate the physics that he desires. Section 2 of the Manual discusses the complexities of simulating realistic problems, including a discussion of the tradeoffs and approximations that must be made. The reader should use this section to learn about defining and interpreting simulations to maximize their meaningfulness. Section 3 covers issues relating to communicating with SOS, including a step- by-step guide to running the code. The central focus of the chapter is a presentation of the SOS Command Language. Also covered is SOS's response to user errors.

Section 4 is a summary of the conventions used in SOS. Many of the commands use the same concept and same format. Section 5 presents a detailed description of SOS commands in alphabetical order. It describes the impact of each command on the simulation, the meaning of each command argument, and the relationships between commands. Section 6 is an annotated sample simulation, including the input file defining the simulation and the output from SOS. Besides being a good example of the use of SOS, the example can be used as a test of proper installation at the local site.

Appendix A is a summary of the SOS commands. It is intended to be copied and kept handy as a quick reference for checking the format for individual commands and for quickly locating commands that perform desired functions.

A detailed description of the mathematical algorithms is available in the SOS Reference Manual on Algorithms.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

SECTION 1

INTRODUCTION

SOS is a three-dimensional, finite-difference code for simulating electromagnetic plasma physics processes, i.e., those processes which involve space charge and/or electromagnetic fields and their interactions.

SOS contains a set of time-domain algorithms such that, beginning from a specified initial state, the code simulates a physical process as it evolves in time. In SOS, the full set of Maxwell's time-dependent equations are solved to obtain electromagnetic fields. Similarly, the complete Lorentz force equation is solved to obtain relativistic particle trajectories, which provide current and charge densities for Maxwell's equations. This approach, commonly referred to as particle-in-cell (PIC), provides self-consistence, i.e., interaction between space charge and electromagnetic fields.

The SOS frequency-domain algorithm analyzes the resonance modes of cavities providing both resonance frequencies and their modal field patterns. The algorithm solves the equation, $\nabla x \nabla x E = \omega^2 E$, derived directly from the full set of Maxwell's equations. The frequency-domain algorithm does not treat particles.

The general approach in SOS relies on the solution of the basic physics equations rather than approximations or simplified models. In addition, SOS has been provided with powerful boundary conditions to represent structural geometries, material properties, incident and outgoing electromagnetic waves, field emission, and so forth. As a result, SOS is applicable to broad classes of self-consistent plasma physics problems, including pulsed-power, charged-particle beams, vacuum electronics devices, semiconductor devices, and others.

µSOS is a subset of SOS without particles. This manual applies equally well to µSOS and SOS, except that any command pertaining to particles is inactive for µSOS. Installed µSOS systems can be upgraded to SOS if the computer hardware includes sufficient disk, memory, and CPU speed. µSOS gets its name from its ability to provide real answers to real problems on a µVAX.

1.1 SIMULATION CAPABILITIES

SOS time-domain algorithms simulate the evolution in time of a defined initial configuration. Phenomena with characteristic time scales ranging from fractions of picoseconds

to tens of nanoseconds have been studied. Typical systems which have been studied with SOS include plasmas, vacuum electronics devices, microwave devices, system generated electromagnetic pulse (SGEMP) effects on satellites, and charged particle and neutral beams. In general, the SOS time-domain algorithm can be used for these and other problems and can study characteristic times limited only by the availability of computer resources.

The SOS frequency-domain algorithm computes cavity response frequencies and their modes. It can compute resonances for cavities with frequencies ranging from hertz to gigahertz. It computes up to the lowest ten or so modes of a device and can compute more modes when symmetries exist in the device. The algorithm has been used to analyze coupled-cavity traveling wave tube (CCTWT) modes and klystron cavity modes including an analysis of the tuning rate of a tunable klystron cavity. Further, the mode information has been used to compute ω - β diagrams and R/Q values. In general, the SOS frequency-domain algorithm can be used for analyzing these and other devices.

SOS has capabilities to simulate in cartesian, cylindrical, and spherical coordinates. Also, SOS can exploit inherent mirror and periodic symmetries in a problem to reduce problem size and to select out a subset of the physics for analysis.

Particle simulation capabilities in SOS are many. SOS handles a variety of particle species simultaneously, including electrons, protons, ions in a variety of charge states, and even arbitrarily defined particles, if desired. Particle motion can be treated relativistically or classically.

Field, photo, and beam emission of particles may be simulated, and in the same simulation. Emitted particles may be of any species. Secondary emission of particles due to the impingement of particles on surfaces is not presently treated.

SOS can start a simulation from many initial states. For time-domain simulations, external magnetic fields may be specified, forces on particles may be specified, and voltages on inlets may be specified. Complex TE and TM wave modes may be injected. For frequency domain simulations, the initial guess at the lowest mode may be specified.

Electromagnetic field emitters, sources, and radiators may be simulated in SOS. Available sources include antennae, cable drivers, and dipole radiators. These electromagnetic field sources may be assigned any pattern of emission over time and even turned on or off at varying times during the simulation. The compliment of field sources, i.e., field pickup antennae and the like, are not currently available in SOS.

Finally, materials with a variety of properties can be simulated in SOS. Perfect conductors can be simulated and are often used to simulate any metal, ignoring the small

resistance in the metal. When desired, specifically resistive properties may be assigned to materials. Air chemistry can be simulated. Also, dielectric and capacitive properties of materials may be simulated, and the dielectric and other relevant properties of such materials may be specified. At the current time, it is not possible to assign a magnetic permeability to a material.

1.2 OUTPUT CAPABILITIES

SOS includes a variety of capabilities to output the results of a simulation. A variety of plots and tables are possible. Also, dumps of the data for further analysis by other codes are possible. Finally, it is possible to output a complete record of the final state of the simulation, and that output can be read in at a later time to continue a simulation from exactly the point where it left off (time-domain only).

Fields are defined to include electric, magnetic, current- density, charge-density, and other fields defined over the simulation for time-domain simulation. It is possible to plot the history of any field at any location in the simulation as a function of time. Such plots can also be transformed to obtain plots of the field amplitude as a function of frequency instead of time. Plots of any field through any one-dimensional coordinate of the simulation can also be created, and for time-domain and frequency domain options. Two-dimensional plots of any field over any two-dimensional portion of the simulation can be output, and options for both time-domain and frequency-domain are also available. Such plots are output as vector plots representing the direction and size of the field at each location by the direction and size of the plotted vector. Any or all of these plots can be output repeatedly at any set of selected times or for any set of selected response modes during the simulation, and for most plots, the same information can be output in the form of printed tables.

For particles, an additional set of plots and tables is available. Phase-space plots showing momentum as a function of position can be created. Trajectory plots showing the position of particles as a function of time can be created. Also, particle statistics and flux tables showing totals for various properties of all the particles in the simulation as a function of time can be generated. These plots can be obtained repeatedly at any set of selected times during the simulation.

There are some outputs which help the user interact with SOS and interpret the results. When SOS reads in a simulation definition, it processes the input, checks it for errors, and outputs its interpretation of the input data along with messages to highlight any inappropriate data. To help the user check that the problem defined by his data is the problem he intended, there are outputs available to show the structure shape, including perspective plots of the discretized devices shown in three-dimensions from any angle. Further, there is a file called a

restart file which can be written at the end of a simulation and used to continue a simulation later. This file can be written out periodically during a simulation as protection against the computer crashing during a long simulation. Also, this file can be made to be written when a simulation in batch mode runs out of its allocated CPU time.

1.3 COMPUTER ENVIRONMENT

The number of computer platforms on which SOS has been installed is continually increasing. The principal computer platforms include: Cray, VAX, MicroVax, SUN Sparcstation, PC486, and IBM 6000. The operating systems include: ultrix and VMS for the VAX's, COS, CTSS, and UNICOS for the Crays; and unix for SUN Sparcstations.

DISSPLA, a commercially available graphics support software product, is required on each local system to obtain graphics output from SOS. Some, but not all, of the plots can be obtained on a line printer if DISSPLA is not available. An interface package for NCAR graphics is also available.

The cost of running SOS is about the same on VAX and CRAY computers or sometimes slightly less on VAX computers, depending upon the exact CPU charges for the particular system. SOS runs about 50 to 100 times faster on the Cray than on a VAX11-780, depending upon the type of problem being run. Typical execution times on the VAX range from a half hour to several days, depending on the type and size of problem and on whether the simulation of particles is involved.

SOS is generally delivered as a set of binary libraries, which is linked to DISSPLA or NCAR at the local site.

SECTION 2

BASIC SIMULATION CONCEPTS

In this section, we present some of the basic simulation concepts which are essential to using SOS. This section will be of interest mainly for the new user, or one who is less familiar with the particle-in-cell method. No attempt is made to be inclusive. Instead, our intent is to present the central concepts and to motivate the use of the code. (Some of the actual SOS commands will be mentioned for cross reference with later sections.) For a more detailed treatment of the PIC method, the reader is referred to the literature. A useful general reference is the text by Birdsall and Langdon.[1]

2.1 TIME

A finite-difference, time-domain (FDTD) simulation follows the evolution of a system in time, beginning from some initial condition. Thus, time is divided into discrete intervals (time steps), which are typically referred to either by physical value (e.g., so many seconds) or by the interval index (an integer). The size of the time step is limited by the various numerical algorithms through stability or convergence (accuracy) requirements. In some cases, it is desirable to use different time steps for different processes in the same simulation. For example, the particle kinematics time step may be an integer multiple of the electromagnetic time (see KINEMATICS and FIELDS commands).

2.2 SPACE

In a finite-difference code such as SOS, it is necessary to divide each of the three spatial coordinates into discrete intervals, in other words, to define the spatial grid (see X1GRID, X2GRID, and X3GRID commands). Each coordinate may be nonuniform in grid size; however, the variation in adjacent grids should typically be limited to 25%. The overlay of grids in the three coordinates produces a large number of approximately rectangular areas referred to as cells. Typically, the spatial aspect ratio of any cell should not exceed a value of five.

---

[1] C. K. Birdsall and A. B. Langdon, Plasma Physics via Computer Simulation, McGraw Hill, Inc., 1985.

SOS provides cartesian, cylindrical, and spherical coordinate systems. In combination with the non-uniform grid spacing, it is often possible to align structural boundaries with cell boundaries, i.e. to have conformal boundaries. When this alignment cannot be attained, then a stair-stepping model of the nonconformal structural boundaries is used.

Sometimes particular structures in the simulation space are so small compared to the rest of the simulation that modeling these structures with grid cells is prohibitive. For such structures, subgrid models are provided. These models approximate the effect of the subgrid sized structure on the fields in the grid cell at which the structure is located and on neighboring cells. This allows analysis of fine detail on relatively coarse grids, saving substantial CPU. The CAPACITOR command provides one such model.

2.3 SPATIAL EXTENSIONS

Because computational expense generally increases with the total number of cells, it is usually advisable to restrict the simulation volume. This implies the need for boundaries, since real devices are not usually electrically isolated from their surroundings. For example, a microwave amplifier device requires a microwave source as well as an extraction port, a waveguide, an antenna, etc. Boundary conditions provide a means of artificially restricting the size of the computational simulation.

In SOS, a complete, continuous outer boundary must be provided around the entire simulation perimeter. This boundary can take many forms. For example, segments of the outer boundary may be composed of perfect conductors (see STRUCTURE command). For a problem involving an ideal cavity, the entire outer boundary would be so defined. More typically, one is concerned with incoming and outgoing electromagnetic waves. Transverse electromagnetic waves defined by the user can be admitted to the simulation (see VOLTAGE command). Similarly, outgoing waves can be simulated (see LOOKBACK command). For outgoing wave applications involving large spatial regions or broadband response, a special resistive boundary can be used (see FREESPACE command). In all cases, there are restrictions on the spatial grid and time step which must be satisfied.

Symmetries can provide very effective boundaries to limit the simulation volume (see SYMMETRY command). For example, periodic symmetry boundaries can be used to study a limited region of a basically repetitive device. SOS offers periodic, axial, and mirror boundaries, and requires that all be explicitly specified by the user. All of these boundary conditions have important implications (e.g., field parity, particle singularity on axis, etc.) which must be understood by the user to ensure a successful simulation.

2.4 ELECTROMAGNETIC FIELDS

Given a discrete representation of time and space, it is possible to write Maxwell's equations in finite-difference form. Many forms are possible. SOS offers three time-domain forms: the centered-difference, time-symmetric, and time-biased algorithms (see FIELDS command); and SOS offers one frequency domain form: the Helmholtz curlcurl form (see the FREQUENCY command). The centered-difference algorithm is explicit numerically and is generally suitable for free-field and some particle simulations. The time-symmetric algorithm is explicit and time-reversible, and is only rarely used for special cases involving energy conservation. The time-biased algorithm is implicit and has the property of suppressing high-frequency noise. It is thus used mainly for particle simulations, which are inherently noisy. The frequency-domain form is used to find resonance modes.

With all of the algorithms, care must be taken that physical frequencies of interest are not suppressed. It is always necessary that wavelengths be resolved by the spatial grid; a good rule of thumb is to use at least ten cells per wavelength. Below six cells per wavelength, the wave cannot be resolved. With the centered-difference algorithm, wave trapping can result. This is manifested in high frequencies with opposite field signs in adjacent cells; of course, this is nonphysical. The time-biased algorithm has the opposite effect - high-frequency fields are damped out and eventually disappear. Unfortunately, the high-frequency fields may be physical, e.g., the output of a microwave device. This damping characteristic of the time-biased algorithm is more difficult to recognize in the simulation; however, it can be predicted from dispersion relations. The frequency-domain algorithm is altered in that the computed frequencies are reduced relative to the physical frequencies by an amount that depends on the number of grid cells used per wavelength.

Each time-domain field algorithm has a limitation in the time step which depends on the grid size. This takes the form of a stability criterion; above the time-step limit, the algorithm becomes numerically unstable and will fail catastrophically. The time-biased algorithm generally has the highest limit; however, it is also more expensive because it is iterative. Thus, centered-difference is always fastest for purely electromagnetic simulations. The stability criterion can easily be calculated given the spatial grid; SOS will also perform this calculation (see COURANT command). There is usually little to be gained by making the time step much smaller than the stability criterion; convergence (accuracy) usually depends on spatial resolution rather than temporal resolution. That is, computed results are usually affected more by the spatial grid size than by the time step.

2.5 PARTICLES

One of the essential concepts in PIC is the macroparticle, i.e., an entity representing large numbers of basic physical particles and having discrete characteristics (e.g., charge, mass, momentum, finite area, etc.). Particles can be created in SOS to represent beam emission, field emission, or photoemission (see EMISSION command). Such particles will be introduced during the simulation (e.g., field emission depends on the dynamics).

Most of the properties associated with particle creation (frequency, particle number, etc.) are specified in the emission models. It is also possible to specify how often kinematics calculations will be done (the kinematics time step), whether the kinematics are nonrelativistic, and so forth (see KINEMATICS command).

The specification of particle creation rates involves some important tradeoffs, since simulation costs are often dominated by particle considerations. Generally, the quality of a simulation can be improved by increasing the number of particles (and increasing the cost). Numerical noise due to particles goes inversely as the square root of the number of particles, so simulation improvement by this means alone can be an expensive proposition. When asked what is the particle per cell density requirement, we usually answer, ten. Of course, this is overly simple.

As with the electromagnetic algorithm, there are time step and spatial resolution requirements for particles. To satisfy the local charge conservation algorithm, particles must not be allowed to transverse more than one cell in a (kinematics) time step. With the stability criterion of the centered-difference field algorithm, this requirement will automatically be satisfied. However, with the time-biased algorithm operating near its limit, the requirement could be violated for relativistic particles. Similarly, field resolution requirements dictate some maximum (say, 25%) cell-to-cell field variation. For example, a space-charge-limited layer might be represented well by 50 cells, barely by 10 cells, and not at all by 3.

2.6 DATA MANAGEMENT

A reality of three-dimensional electromagnetic codes is their need for substantial computer memory. A 100 x 100 x 100 cell problem requires 10^6 words per field plus data space for many thousands of particles. Further, while more and more large computers have memory to handle these requirements, many of the fastest machines still do not. Thus, SOS allocates memory and uses disk when the memory allocation is exhausted.

The important feature of this to a user is that data is divided into planes according to their i_3 coordinate. Thus, the code has a few special properties regarding i_3. First, defining the problem to minimize its extent in i_3 reduces the number of planes of data, generally increasing processing speed. Second, a periodic boundary cannot be defined along i_3. Other than these few things, the data management is transparent to the user.

2.7 PRACTICAL CONSIDERATIONS

The serious user of a code such as SOS will encounter cost limitations in his quest for simulation integrity and accuracy. A reasonable objective might be to achieve the highest quality consistent with time and cost limitations. To aid in this process, a cost algorithm is essential. This algorithm, which can be developed for any computer, is best arrived at by empirical fits to actual simulation cost data. It is simply an equation which expresses CPU (or dollar) costs as a function of simulation variables such as cell number, average particle number, time steps, algorithm choices, etc. This cost algorithm will assist the user in making the tradeoffs essential to the necessary compromise.

This page is intentionally left blank.

SECTION 3

USING SOS

Running SOS requires four basic steps. First, one must create an input file containing a description of the problem to be simulated. Once the model has been quantified, the relevant commands can be selected and placed in a file using a text editor or using the interactive front end (IFE) in SOS. Often, the input file can be created by modifying a previous input file. Second, one must run SOS, which reads the input file, performs the numerical computations, and writes some output files. SOS is typically run as part of a batch job, which also retrieves the input file and disposes of output files. If SOS detects an error, it will stop before the simulation starts; one must then examine the output, correct the input, and rerun SOS. In any case, critical examination of the processed input data is essential to ensure simulation integrity. Third, one must print the output file containing simulation results. If plotted output has been requested in the input, then one must plot the resulting graphics metafile on some graphics device using the system-supplied postprocessor. In the fourth (optional) step, one may continue the simulation using the restart capability of SOS.

3.1 COMMAND LANGUAGE

The command language used in SOS is the same as the command language for other members of the SOS family of codes. The commands vary from code to code, but the language is the same.

The basic command format is

```
<command name> <argument> ... <argument>  ;
```

The fixed aspects of the format are the command name and the semi-colon (;). The command name must be fully specified in the appropriate upper or lower case (machine-dependent, upper case on VAX). The semi-colon (;) indicates the end of the command and is required since the number of arguments may be variable. A single command may be split over several input lines. Also, several commands may be placed on a line. Command arguments must be entered in the order specified in the command description.

The number of arguments in a command varies from command to command. Some commands take no arguments; some accept as many as 80. Some commands take a fixed number of arguments, while others accept a variable number. Sometimes arguments are optional or may

be repeated. In certain commands, the value of one argument can determine the format for the rest of the command.

There are four types of arguments: alpha, integer, real, and strings. Arguments are also classified by usage, including arithmetic expressions, functions, and variables. Expressions are a type of string; functions are a type of alpha argument; and variables have the properties of alpha, integer, real, and string arguments, depending on their usage.

3.1.1 Alpha Arguments

A simple set of rules for creating alpha arguments is to begin them with a letter A-Z, to include only letters and numbers in the argument, and to restrict the number of characters to 16 or less. These simple rules can be violated, but then the rules become complicated.

Generally, no special characters that are used for parsing the line are allowed in alpha arguments (e.g., double quote (" "), comma (,) and semi-colon (;)). To begin with anything besides a letter or to include a comma, a space, or a semi-colon, the argument must be enclosed with double quotes (e.g., "1 2p" or "@*DJ/"). The only character that an alpha string may never contain is a double quote (" "). In arithmetic expressions (see Section 3.1.5), alphas must begin with a letter. Alphas in expressions must be enclosed in single quotes to include special characters needed in parsing expressions, (e.g., comma (,), single quote ('), space ( ), left and right parentheses ([and]), asterisk (*), slash or divide (/), plus (+), and minus (-)). Function names (see below) that include special characters cannot be used in expressions. The 16 character length restriction cannot be violated.

Alpha arguments are used for specifying options and parameter names in input commands. They are also used for specifying variables (see Section 3.1.5) and functions (see Section 3.1.6) Some alpha arguments allow the user to select from a fixed list of options. The set of options may be described as a convention as in Section 4 or in the command. In such cases, the input alpha argument need only contain enough letters to uniquely identify the option desired. Thus, if the options are ON, OFF, and LIST, then inputting just L is equivalent to inputting LIST.

3.1.2 Integer Arguments

Integer arguments are numbers that are, to within machine precision, integers. Some commands, such as the DEFINE and DO, will truncate reals to integers when necessary. Others will detect an error when a non-integer number is input, and others will simply do the wrong

thing if the wrong data is input. In any position in a command in which an integer argument is requested, an integer variable may be used.

3.1.3 Real Arguments

A real argument is any number that fits within the range for a real number on the computer being used. In any position in a command in which a real argument is requested, a real or integer variable may be used. The integer variable will be automatically converted to a real variable.

3.1.4 String Arguments

Strings are character strings that may contain special characters such as commas, spaces, and semi-colons. Also, the character strings may be longer than 16 characters; the maximum length of the strings is command-dependent. Strings are always bracketed by double quotes to delineate them. Therefore, a double quote cannot be placed into a string. Strings may span several lines.

Arithmetic expressions are a particular usage of strings. These expressions use FORTRAN style to express a computation to be performed. The expression evaluator understands almost any expression that would be understood by a FORTRAN compiler. Expressions may extend to multiple lines in a free format. Because they are strings, they must be enclosed in double quotes. For more information about arithmetic expressions, see the DEFINE and FUNCTION commands in Section 5.

The values of variables may be inserted into strings using single quotes to identify the variable in the string (see Section 3.1.6 for details). Because the single quote is used as a delimiter in strings, it cannot be simply inserted as a character into a string.

3.1.5 Variables

Variables are defined using the CONVERT, DEFINE, and DO commands. They have a name and an associated numerical value. Variable names have the same rules as alpha arguments and the same recommendation that the variable name begin with a letter A-Z, and include only letters and numbers. They must be 16 characters or less in length. There are integer variables (any variable beginning with a letter I-N, and there are real variables (any variable beginning with a letter A-H or O-Z).

Variables may be used in commands in place of any real or integer argument. Variables may be used in arithmetic expressions including the definition of other variables. They may also be used in the input strings. To insert the value of a variable into a string, the variable is enclosed in single quotes. These identify it as a variable whose value is to be substituted into the string.

The specification of a format for the insertion of the number is optional. The format is appended to the variable inside the single quotes in the string. The variable and the format are separated by a colon (:). The formats are given in standard FORTRAN style. If no format is given, then for integer variables, the number is inserted using as few characters as possible, and for real variables, the number is output in E format with 3 places to the right of the decimal.

Variables are often used as a way to document the meaning of important numbers. They are also used to reduce the number of places in the input in which a number must be changed if a value of a parameter changes. An example of a portion of input using variables is:

```
DEFINE FREQ  1.33E8 ;     ! Set a frequency
DEFINE PI 3.1415926 ;     ! Define the number pi

! Include the frequency in the title in megahertz

DEFINE "FREQM= FREQ/1E6"  ;
TITLE "Device driven with sine wave at
        'FREQM:F6.2' Mhz";

! Define a sine function at the frequency

DEFINE "OMEGA=FREQ*2*PI"  ;
FUNCTION "SINWAV(T)=SIN(OMEGA*T)" ;
```

Note the use of the exclamation point (!) to set off comments. Any data on the same line following this symbol is ignored and is not processed as input data.

3.1.6 Functions

Many commands require as input a specification of a function that provides a dependent value of one or more independent variables. These functions are created and given a name using the FUNCTION command, and they are then referenced in the commands where the specification of a function is required. The definition of the function allows the generic

specification of a multi-variate function; the meaning and order of the independent variables to the function depend upon the usage in the command where it is referenced.

Functions defined via expressions are not currently recognized in subsequent expressions. Also, no function is recognized in an expression occurring in a DEFINE command.

3.1.7 Defaults

Many of the commands have defaults specified. Such commands may be omitted from the input, in which case the default values indicated will be used by the code.

3.1.8 Optional and Repeat Arguments

For many commands, some or all of the arguments are optional. Such arguments are specifically identified with the commands. Such arguments can be omitted from the input for the command. Sometimes several arguments are optional as a set. For such sets, if a value for one member of the set is specified, then values must be specified for the entire set of arguments.

Some commands allow or require the repeat of an argument or a set of arguments in a command. Such arguments must be included at least once.

3.2 PROGRAM OPERATION

SOS can be run from a terminal or submitted as a batch job. The appropriate machine-dependent command starts the SOS execution.

During batch execution, there is no communication by the user to SOS. SOS reads INFIL.DAT (VMS) and INFIL (UNIX and DOS) on the local file directory (unless overridden by the assignment of logicals using the operating system) to obtain all the information it needs. SOS puts basic output into OUTFIL.DAT (VMS) and OUTFIL (UNIX and DOS) on the same directory. It is possible on some computers, including the VAX, to watch the progress of the run as it executes by assigning OUTFIL.DAT to the log file and printing or typing that log file at any time.

During operation from a terminal, SOS will locate and read the input file (if it exists), and then (whether or not it exists) SOS will ask whether or not an interactive session is desired. If not, answer 'N' and program execution will proceed as for a batch job except that a terminal is

connected to the job. To watch the progress of the run during an interactive run, use assignment of logicals, if available, to assign OUTFIL.DAT to the terminal. This will send output to the terminal, and therefore OUTFIL.DAT will not be recorded anywhere on disk for later printing.

To have an interactive session, answer 'Y' and define your simulation as described in Section 3.1.3. OUTFIL.DAT may be assigned to the terminal or not as before.

3.2.1 Command Format Error Checks

SOS reads input one command at a time and processes that command. It checks for many possible format and data errors in each command and marks and reports those errors. Then SOS stores internally the data provided in that command. The data is not actually used internally until a START or STOP command is read. SOS processes all commands and lists all errors in the file OUTFIL.DAT. Bad data is indicated with a series of question marks immediately following the data in the output. If there are any command format errors, SOS will terminate after processing all commands and will not run the defined simulation.

For each command, SOS checks for a valid command name. The entire command name must be input -- no abbreviations are allowed. SOS checks that the correct number of arguments have been entered and that each argument is of the right type (numeric or alphanumeric). In addition, SOS differentiates integers from non-integers. All these checks help guarantee that the commands are correctly used and that the simulation is defined as intended. SOS also checks that numbers are in specific ranges and that alphanumeric arguments are valid. SOS does not, during this phase of program execution, do any range checking that would imply a specific ordering of commands. Thus, for instance, grid indices are checked for being above 1, since that is always required, but they are not checked against the defined range of indices as specified in an X1GRID command. For alphanumeric options, SOS allows abbreviation as long as the abbreviation is unique within the context of the command. Ambiguous specifications are errors. Thus, the symbol Y or N is sufficient when the alphanumeric option is YES or NO, while the symbol A is ambiguous and therefore an error when the option is ALIGNED or ANTI-ALIGNED. In this case, the symbol AL or AN is sufficient. Abbreviations are not accepted for the command keywords themselves.

SOS will accept a certain number of repetitions for each command. If the input file contains more input than SOS can handle, an "Input Storage Exhausted" error results. If such an error occurs and truly reflects the complexity and particulars of the desired simulation, then SOS will have to be regenerated from source code with enlarged array allocation. When SOS finds a START or STOP command, it outputs a summary of the errors encountered up to that point. If there are any errors in the input, execution is terminated.

3.2.2 Simulation Consistency Checks

When SOS encounters a START command and there have been no command format errors, checks are then made on the meaningfulness of the data. This second set of checks is intended to check for consistency between commands. Once again, if any errors are found execution is terminated.

The simulation consistency checks include tests for the following errors:

spatial coordinate arrays exceeded,
field arrays exceeded,
Courant stability criterion exceeded, and
presence of function definitions for specified functions.

SOS checks for these errors and lists all that have occurred. If any have occurred, the simulation is terminated at this point. If no errors have occurred to this point, SOS will attempt to run the simulation.

3.2.3 Helping SOS Check the Simulation

SOS is not yet completely user-error resistant. There are many checks that the user must make because SOS does not make them. For instance, a continuous outer boundary must be specified. Failure to do this will cause uncertain simulation results, yet SOS does not make this check. Another example of the many checks not made is whether antennae are placed in physically meaningful locations. An antenna placed accidentally inside a conductor instead of in a cavity will not radiate. SOS will still run, but its results will usually not be meaningful.

To guard against such possible errors, it is advisable to run SOS for a few time steps at first and to output many plots that can show whether the simulation is at least basically running correctly. For instance, if an antenna is accidentally placed where it cannot radiate, then this error may show up quickly, because the electromagnetic fields will remain zero.

SOS relies on the detection of certain other errors during the process of trying to use the input data. There are many such errors that are possible. SOS will output a message indicating the presence of an error. Thus, to minimize CPU time wasted, it is beneficial to run the simulation first for a few time steps and then to scan the output for error messages.

3.3 GETTING OUTPUT

SOS produces a variety of disk files containing the results of the simulation, including both printed and graphics output. These files are written to disk storage during the simulation, so output may be observed immediately afterward or later.

3.3.1 Printed Output File

SOS always produces an output file suitable for printing on a standard printer. Specifically, it writes a Fortran formatted file of records limited to 132 columns to OUTFIL.DAT. This file contains several types of information.

First, it contains processed input, that is, one printed line for each argument of each command in the input file, along with the definition of each argument and any error messages. SOS does some checking of the input file, and if it finds errors, it prints error messages with the processed input. Subsequent termination allows an opportunity to make changes to the input file before incurring the expense of a simulation.

Second, it contains information printed periodically as the simulation progresses: particle statistics, simulation energy, tables of field values, plots rendered in lineprinter graphics, etc. All such information is written to the output file only if the associated commands are present in the input file. Also, errors detected during the simulation will result in error messages in the output file.

Third, it contains a summary of performance-related parameters, e.g., number of grid cells, number of time steps, number of particles processed, and number of plots produced. This information can be used to estimate how much CPU time a simulation will require.

Fourth, if requested, it contains time histories of all simulation variables defined by OBSERVE commands. This output is rather voluminous, but can be useful for comparing with other simulations.

3.3.2 Graphics Output File

SOS may be linked to run with NCAR or DISSPLA. When system graphics is selected with the GRAPHICS command, SOS writes all of its plots to a graphics metafile, the name of which depends on the system. After the simulation, this file must be read by the system-supplied

NCAR or DISSPLA postprocessor to produce plots on a particular graphics output device. Also, the metafile can be stored for later use or used to produce multiple plots on different devices.

In case neither DISSPLA nor NCAR graphics are available to SOS, graphics may be rendered as lineprinter graphics when so instructed by the GRAPHICS command. Plots will be displayed with low-resolution and written to the output file.

3.3.3 DUMP Output Files

SOS can produce output disk files for use with the POSTER post-processor. The types of information which may be included in these files includes the simulation title, spatial-grid values, conductor, symmetry and other boundaries, and field and particle information. The files may be created in binary for compactness or in ASCII for easy transport of the data between computers. The data is sorted into three files, 'FLD' for field data, 'PAR' for particle data, and 'GRD' for grid and other miscellaneous data. The DUMP command provides controls for which types of data are output, the output format, and the names of the files created. The DUMP command further interacts with the output specified by SOS output commands such as CONTOUR, OBSERVE, PHASESPACE, and RANGE to determine more specifics about which data to output to the DUMP output data files. Basically, any data that can be plotted by SOS can be routed to a DUMP data file. After the simulation, the DUMP files can be read into POSTER for further processing and graphical display, including motion pictures.

3.3.4 Restart File

If requested through the RECORD or TIMEOUT commands, SOS will produce a restart file. This is a binary file that contains the entire state of SOS at some point in a simulation. With the RESTART command, SOS will read such a file and continue the original simulation at the point when it was stored. This file is a Fortran unformatted file whose name is specified by the RECORD or TIMEOUT commands.

3.4 CONTINUING THE SIMULATION

An SOS simulation may be continued in a separate execution by means of the restart feature. The complete state of the simulation may be stored as a disk file at a particular time step; later, this file can be retrieved and the simulation continued at that time step. The restart feature helps safeguard against the following events:

repeating an entire simulation to extend time,
failure due to a CPU time limit,
computer failure, and
inadequate output specification.

3.4.1 Creating the Restart File

A restart file can be written several times during a simulation with the RECORD command. This command specifies both the name of the restart file and the time steps at which it is to be written. During the simulation, a restart file containing the simulation state at the current time step is written on permanent mass storage; if written again later in the same simulation, the previous restart file will be overwritten.

A restart file can also be created with the TIMEOUT command. This command allows SOS to finish gracefully and write a restart file when a CPU time limit threatens to end the simulation prematurely.

3.4.2 Restarting the Simulation

A simulation previously saved as a restart file may be continued with the RESTART command. To restart, one must place this command at the beginning of an input file. As SOS processes the input file, the specified restart file is read, and the simulation is restored to the state at which the restart file was written. Even the memory of the original input data is restored, so that commands following RESTART will simply add to or supersede the original commands. When SOS processes the START command, the simulation begins with the time step following the one at which the restart file was written.

3.4.3 Modifying the Original Simulation

When restarting a simulation, SOS restores the state of that simulation, processes the remaining commands in the input file, and then commences execution of the simulation. The remaining commands are treated as though they were appended to the original input file; they either supersede, change, or add to the original commands. The most common application of this property is to supply a FIELDS command to extend the simulation in time. Another particularly useful application is to complete a simulation with relatively few graphics, add a flurry of graphics commands, restart, and run for only the few time steps needed to produce the additional graphics.

Some commands simply overwrite their previously specified values. Others, including most graphics commands, add to information as if they had been present in the original input file. All remaining commands, however, specify arguments that cannot or must not be changed during the simulation. These commands cannot be used when restarting.

This page is intentionally left blank.

SECTION 4

CONVENTIONS

Many of the commands in SOS use arguments in the same basic format. For example, various output commands usually specify a field variable as well as spatial and temporal indices. Thus, these arguments are written according to conventions which are common to all commands. These conventions are identified and described in detail only in this section of the Manual. Section 5, which describes commands in detail, will refer to the conventions by name only, without elaboration.

4.1 SYNTAX

The command descriptions in Section 5 use a concise style to describe the format of each command. Upper case indicates portions of the command that must be typed in exactly as shown. Symbols representing the input arguments are shown in lower case. Generally, the symbol reflects the meaning of the data; for example the symbol, afield, appearing in a command description requests the input of field information. Certain qualifiers appearing in symbols provide additional information about the argument required. For example, the prefix a in afield reveals that an alpha argument is to be supplied.

4.1.1 Argument Qualifiers

Certain symbol qualifiers indicate the required type of argument, as follows:

Argument Type	Symbol Qualifier
alpha	symbol begins with an 'a'.
integer	symbol begins with an 'i, j, k, l, m, or n'.
string	symbol enclosed in double quotes.
real	symbol not beginning with 'a', 'i-n', or ".

Other symbol qualifiers indicate a specific usage of the argument, as follows:

Argument Use	Symbol Qualifier
function	parentheses within the argument name.
function/real	parentheses within the name and an '*' at its end.
expression	equal sign within the double quotes of the string.
optional	brackets enclosing the arguments(s).
repeat	'[,...]' following the argument(s).

Of these qualifiers, only the double quotes and the equal sign generally appear in the actual input. The constraints on the actual input are discussed in Section 3 for each argument type.

As an example, the command format

```
DEBUG ON [aopt] ;
```

means that "DEBUG ON" is typed in and that a further specification is optional. The user does not type in aopt or [aopt] but one of a set of possible options.

4.1.2 Function and Function/Real Qualifiers

Functions are identified by the prefix 'a' in the name (because they are an alpha argument) and by '( )' embedded in the name. The parentheses are used to specify the order and meaning of independent variables of the function. For instance,

```
avoltage(phase)
```

requests a user-defined function for a voltage as a function of phase.

In some cases, a function or a real value may be input. An asterisk (*) at the end of the name indicates this possibility. For example,

```
afield(phase)*
```

indicates that either a function name describing a field as a function of phase may be input, or if the value is independent of phase, then a simple number may be input.

The user does not input the parentheses or the word 'phase' or the '*' in the actual command for the above examples.

4.1.3 Arithmetic Expression Qualifier

Arithmetic expression arguments in command syntax are identified by an "=" sign in the argument name. For instance, an expression argument might appear in a command definition as

```
... "aname=expression" ...
```

The representation shows that the argument is an expression, and it also shows that the result of the computation is equated to an alpha argument aname. The command will then go on to explain the meaning of aname and the expression in the context of the command.

Examples of expressions that may be valid in a command are

```
"RESULT = (X**2 + Y**2 ) ** (.5) "
"FUNCTION(A,B) = SIN(ATAN(A+B)) ".
```

The quotes and "=" sign must be present in the expression in the input file.

4.1.4 Optional and Repeat Argument Qualifiers

In some commands, arguments are optional or may be repeated. Optional arguments are placed in brackets '[]'. These arguments may be omitted. The brackets are not included in the input. If more than one argument is enclosed in a single pair of brackets, then the entire set of arguments must either be omitted or included and in the order shown.

Repeat arguments are arguments that must be included at least once and which may or must be included more than once. Repeat arguments are also placed in brackets '[]'. For example,

```
ktime [,...]
```

means that at least one, but possibly more, integer values must be included in the input. The example,

```
[ktime[,...]]
```

means that all of the arguments are optional. As with optional arguments, the brackets may enclose more than one argument, in which case the entire set must be repeated, if any element of it is.

4.2 FIELD VARIABLES

4.2.1 Physical Units

Input and output for SOS uses the rationalized mks system of units. The only exception is the definition of generalized particle momentum, which omits rest mass. Commonly used variables, definitions, and units are listed in Table 4.1. In all cases, the correct physical units are stated in the input variable definitions and in all printed and plotted output.

4.2.2 Generalized Coordinates

For simplicity, the code uses generalized coordinates (x_1,x_2,x_3) to describe three-dimensional space. The convention identifying generalized coordinates with physical coordinates for the two coordinate systems available in SOS is:

Physical Coordinate System	Generalized Coordinates (x_1,x_3,x_3)
cartesian	(x,y,z)
cylindrical	(r,θ,z)
spherical	(θ,ϕ,r)

4.2.3 Alphanumeric Labels

Many command arguments consist of alphanumeric labels. These fall into two categories: user-defined and code-defined. User-defined alphanumerics provide flexibility in writing input data. For example, the user is required to provide a unique label, or name, for each separate transmission line that he enters. This provides the means to request current or voltage output from a particular line, or to match a particular line to the two-dimensional fields. Code-defined

variables allow the user to refer to specific fields which are already in use by the code. In the command format, a field argument is identified by the symbol,

```
afield.
```

Upon encountering this symbol, the user must supply an alphanumeric argument for an electromagnetic field, e.g.,

```
B3.
```

(Note that the first letter in afields implies an alphanumeric response.) The complete set of alphanumeric labels the for electromagnetic fields is given in Table 4.2. Labels for strut model variables and energy balance components are given in Table 4.3.

4.3 TIME VARIABLES

4.3.1 Time-Step Index

In SOS, time is measured by discrete intervals,

$$t = \mathrm{ktime} \cdot \delta t_f \ ,$$

where ktime is an integer and δt_f is the electromagnetic field's time step.

Thus, the physical variables are defined at discrete values of time. Figure 4.1 illustrates the general definition of field variables in time. A simulation always begins from time, t = 0, which corresponds to a time index, 0. At this time, the initial state of the simulation is computed. The initial state may be simple (e.g., a field- and charge-free state) or complicated (e.g., a space-charge and electrostatic field distribution). Whatever the initial state, it is computed at t = 0.

For simulations which involve multiple runs, i.e., an initial simulation followed by one or more restarts, the time scale is permanently determined by the first simulation. Thus, if the user desires to extend a simulation in time by means of a restart, he must refer to and use the time scale and zero established in the first run.

Table 4.1. Physical units.

Variable	Definition	Units
E	electric field	V/m
B	magnetic field	W/m^2
J	current density	A/m^2
ρ	charge density	C/m^3
ϕ	scalar potential	V
I	transmission line current	A
V	transmission line voltage	V
q	generalized coordinate	m or rad
p	generalized momentum	m/sec
t	time	sec
θ	azimuthal coordinate	rad
r	radial coordinate	m
x,y,z	cartesian coordinates	m

Table 4.2. Alphanumeric labels for electromagnetic fields.

Alphanumeric Label	Physical Symbol	Physical Definition
E1 E2 E3	E_1 E_2 E_3	electric field
B1 B2 B3	B_1 B_2 B_3	magnetic field
J1 J2 J3	J1 J2 J3	current density
Q0	ρ	charge density
E1AV E2AV E3AV	E_1^a E_2^a E_3^a	average electric field
B1AV B2AV B3AV	B_1^a B_2^a B_3^a	average magnetic field

Table 4.3 Alphanumeric labels for other variables.

(a). Strut model variables.

Alphanumeric Label	Physical Symbol	Physical Meaning
CURRENT	I	transmission line current
VOLTAGE	V	transmission line current

(b). Energy balance components.

Alphanumeric Label	Physical Definition
CREATED	energy introduced by creation of particles
DESTROYED	energy lost by destruction of particles
SURVIVING	energy of existing particles
GAINED	energy given by fields to particles
VOLTAGE	energy lost through voltage boundaries
LOOKBACK	energy lost through lookback boundaries
MATCH	energy introduced through transmission lines
BOUNDARY	energy lost through all boundaries
ELECTRIC	energy of electric fields
MAGNETIC	energy of magnetic fields
EM	energy of both electric and magnetic fields
TOTAL	energy of particles and E/M fields
RESIDUAL	energy not accounted for
RATIO	residual-to-total ratio

time	$(k-1)\delta t_f$	$(k-1/2)\delta t_f$	$k\delta t_f$	$(k+1/2)\delta t_f$	$(k+1)\delta t_f$	$(k+3/2)\delta t_f$
particle coordinates (x) and momenta (p)	x^{k-1}	$p^{k-1/2}$	x^{k}	$p^{k+1/2}$	x^{k+1}	$p^{k+3/2}$
current (J) and charge densities (ρ)	ρ^{k-1}	$J^{k-1/2}$	ρ^{k}	$J^{k+1/2}$	$\rho^{k+1/2}$	$J^{k+3/2}$
electromagnetic fields	$\mathbb{E}^{k-1}$	$\mathbb{B}^{k-1/2}$	$\mathbb{E}^{k}$	$\mathbb{B}^{k+1/2}$	$\mathbb{E}^{k+1}$	$\mathbb{B}^{k+3/2}$

Figure 4.1. Temporal definition of fields.

In the command format, a time-step index is identified by the symbol,

```
ktime.
```

In response, the user must supply an integer representing the value of the index. (Note that the first letter in the symbol, ktime, implies an integer response.)

4.3.2 Time-Step Multiple

It is also possible for processes to occur at multiples of the electromagnetic time step. For example, the particle kinematics time step, δt_k, can be an integer multiple of δt_f, or

$$\delta t_k = \texttt{kmultiple} \cdot \delta t_f \cdot$$

Figure 4.1 illustrates the relationship between electromagnetic field and particle kinematics time steps for the case, kmultiple = 1.

In the command format, a time-step multiplier argument is identified by the symbol,

```
kmultiple.
```

In response, the user must supply an integer.

4.3.3 Time-Step Sequence

A timing argument is included in many commands to specify a sequence of time steps at which some event takes place. For example, graphic-output commands use this argument to specify when, during the simulation, the existing data should be plotted. The timing argument is identified by the symbol,

```
atimer.
```

This is an alphanumeric label which must be defined by the user in a TIMER command.

In the TIMER command format, time-step sequence arguments are identified by the symbols,

```
DISCRETE ktime [,...]
PERIODIC kstart kstop [kstep].
```

For the discrete option, a complete list of time steps in the sequence is input. This is useful when the time steps are scattered aperiodically in time. When the periodic option is used, three integers are input to specify the periodicity: a beginning time step, an ending time step, and a time step interval. This option functions like a Fortran do loop except that the time step must be positive. The input of a time step is optional. The default value for kstep is one.

For example, two valid sets of time-sequence arguments are

```
DISCRETE 100 200 300
PERIODIC 100 300 100.
```

Either would produce the same response: the event (e.g., graphic output) would occur at time-step indices of 100, 200, and 300.

4.4 SPACE

4.4.1 Cell Indices Convention

The three-dimensional volume is divided into a large number of spatial cells. That is, each of the three coordinates (x_1,x_2,x_3) is divided into finite lengths, designated by full-grid locations (e.g., x_1^f). Thus, the unit cell is bounded by full-grid locations on all six sides. There are also defined half-grid locations (e.g., x_1^h) located near the center of the cell. Each of the electromagnetic fields is defined precisely with respect to this finite grid using the "well-centered" approach, as illustrated in Figure 4.2. For example, the x_1 -component of electric field (E_1) is defined at the half-grid location in one coordinate (x_1^h) and at full-grid locations in the other (two (x_2^f and x_3^f). The current density fields (J_1,J_2,J_3) are spatially coincident with their electric field counterparts (E_1,E_2,E_3). All averaged electric and magnetic fields are defined at the corner of the unit cell (full-grid locations in both coordinates). Charge density and scalar potential variables are also defined at the corner of the unit cell.

To specify the spatial location of any particular field, the user can simply state the appropriate cell indices (i_1,i_2,i_3). In the command format, cell indices arguments are identified by the symbol,

```
ipoint .
```

(Note that the first letter in this symbol implies an integer response.) Upon encountering these symbols, the user must supply three integers which specify the cell.

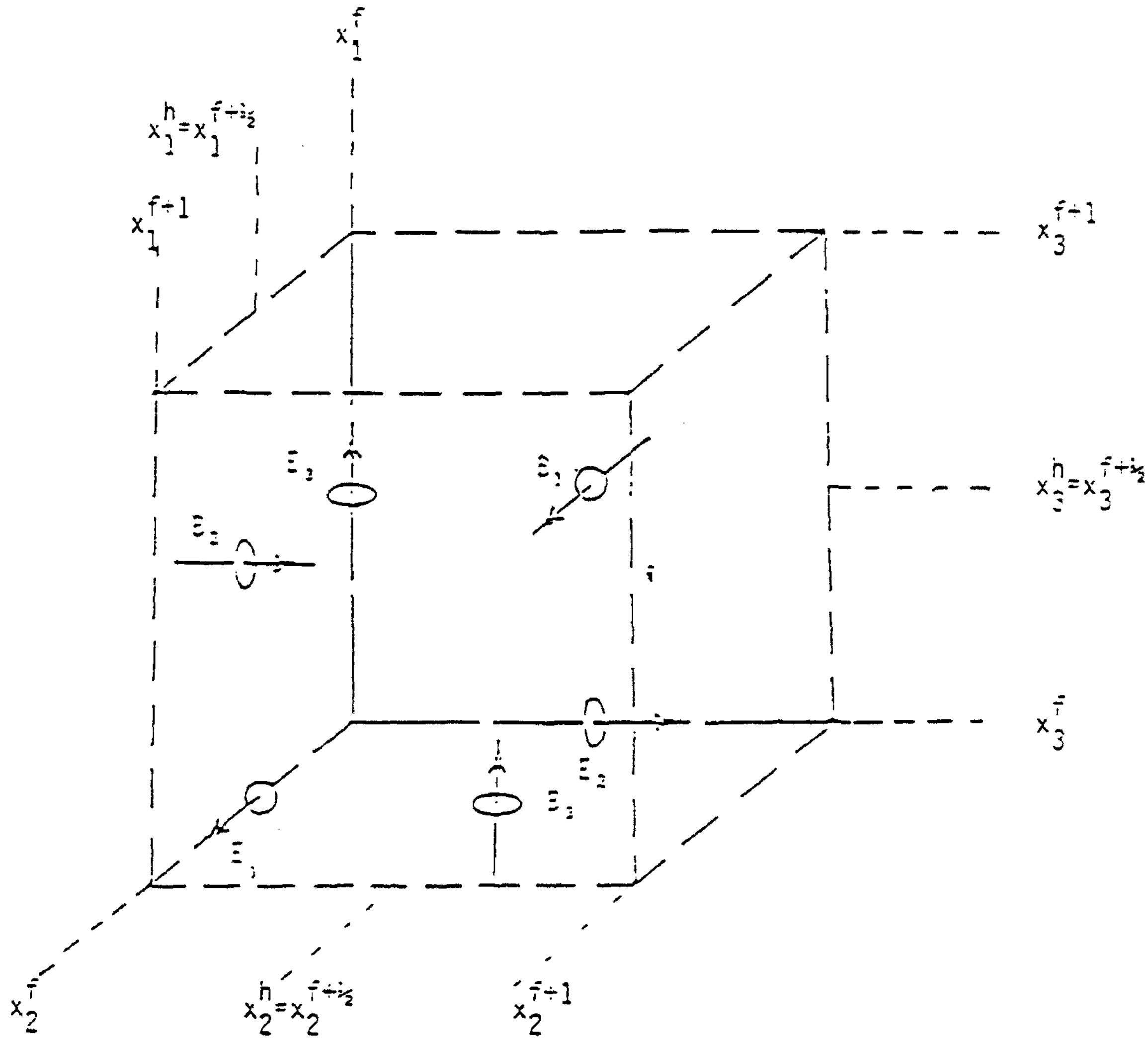


Figure 4.2. **Spatial definition of well-centered fields on sample grid cell. Superscript f is full-grid point. Superscript h is half-grid point .**

The cell indices arguments are commonly used with an alphanumeric argument. For example, if the command format includes the symbols,

```
afield ipoint,
```

then a valid response might be

```
E1 23 47 10,
```

specifying the electric field E_1 at cell indices (23,47,10).

4.4.2 Line Segments

In many cases, it is not sufficient to specify a single spatial location; instead, a spatial range is required. Examples of such a requirement might involve

integrating a field over space, and
plotting a field as a function of space.

Such input data is described using line segments. Each line segment is specified by spatial indices representing the end points of the line. Specifically, the spatial indices refer to full-grid point locations representing the (inclusive) extremes in each coordinate. The number of discrete fields defined within this range depends upon the character of the field component, i.e., whether it is defined at half- or full-grid locations. Line segments must be conformal with the spatial grid. That is, for two dimensions, the low and high grid range values should be identical, while for one dimension the low and high range may include several grid points. Thus,

```
1 10 3 3 22 22
```

specifies a line extending from grid point 1 to grid point 10 in x_1 and positioned at gridpoints ($i_2 = 3$, $i_3 = 22$).

Line segment arguments are represented in the command format by the symbol,

```
iline .
```

4.4.3 Surfaces

Several commands require specification of a two-dimensional spatial surface. The command format is,

```
isurface.
```

Again, six integers are required, representing the extremes in each of the three coordinates. The area must have zero thickness in at least one dimension, i.e., the surface must be conformal with one direction. In some cases, it may allow zero extent in two or more dimensions.

4.4.4 Surface Normals

For many commands (SYMMETRY, VOLTAGE, LOOKBACK, . . .), it is necessary to specify an orientation to a surface. This is done by specifying the relative direction between a unit vector normal outward from the surface and the direction of increasing values of the conformal coordinate. If the unit vector points towards increasing values of the conformal coordinate, then the orientation is aligned. If the unit normal points in the direction of decreasing values of the conformal coordinate, its orientation is anti-aligned.

In the command format, the surface normal argument is represented by the symbol,

```
asnor.
```

Valid responses are the alphanumeric labels,

```
ALIGN or ANTI-ALIGN.
```

The surface normal argument is most often encountered with a surface, i.e.,

```
asnor isurface.
```

4.4.5 Volumes

Some commands require the specification of a volume. In the command format, this is represented by the symbol,

```
ivolume.
```

Once again, six integers are required, representing the extremes in each of the three coordinates. Zero thickness in one or more dimensions may be allowed for some commands.

4.5 GENERALIZED ITERATIONS

The different interpretations of iteration, depending upon the algorithm used, necessitate several meanings for iteration. In the centered difference algorithm, an iteration and a time step are synonymous. For the time-biased algorithm, they are not. For the frequency-domain algorithm, there are iterations to find each resonance, and then there are iterations over resonances to be found.

4.5.1 Step

A step is defined to be a time step for the time-domain algorithm and the finding of a resonance for the frequency-domain algorithm. Any input variable with both these meanings is identified as a step in the command definition. Other types of iteration input (not including input that is specifically time-step) is identified as an iteration, and its meaning is specific to each command. To summarize, this is represented in command formats by symbols of the form,

itime - time step
astep istep - generalized step
iter - other iteration input.

4.5.2 Step Interval

Step-interval arguments are included in many commands to specify a sequence of intervals at which some event takes place. For example, graphic output commands use these arguments to specify when, during the simulation, the existing field data should be plotted. The timing argument is identified by the symbol, atimer. This is an alphanumeric label which must be defined in the TIMER command.

In the TIMER command, step-interval arguments appear as a variable number of consecutive arguments within a command. The first argument determines the type of interval specification, periodic or discrete. For the periodic case, the remaining three parameters are, in order, a beginning step index, an ending step index and a step interval. This case functions like a Fortran do-loop; the event occurs at the beginning step and then repeats at the specified interval

until the ending step index is exceeded. For the discrete case, the next argument specifies the number of remaining arguments, which are the step indices at which the event should occur.

SECTION 5

COMMAND SYNTAX

This section describes each SOS command in detail. It gives the function of the command, the command format, and the order and meaning for each argument in the command. When there are multiple possible formats for a command, each format is listed and described. The commands are listed in alphabetical order by command name.

BALANCE

Function: Specifies frequency and range of electromagnetic and particle kinetic energy calculations.

Format:

BALANCE istep ivolume ;

Arguments:

istep - step interval (integer).
ivolume - volume for which to compute field energy (integers).

Description:

The BALANCE command controls the energy calculation performed by SOS. It may be used to see how well the total energy present in the simulation tracks energy sources and sinks. It may also be used as a test of energy conservation, though the calculation of total energy isolates individual contributions which can be viewed through the OBSERVE command. Also, a summary of these energies is printed during the simulation once every istep time steps.

SOS calculates energies and fluxes whenever the BALANCE command is present in the input file. For particles, kinetic energies of created particles are accumulated from the beginning of the simulation, as are kinetic energies of destroyed particles.

The total, instantaneous kinetic energy of all particles in the simulation is calculated, but only once every istep time steps due to the expense of the calculation. From these energies, the energy gained by particles is accumulated from the beginning of the simulation. Four components of particle energy are accessible via the OBSERVE command. These energy components are first calculated separately for different particle species and then summed over species, so information concerning a particular species may be analyzed.

For electromagnetic fields, both total energy and lost energy are calculated. Only the three-dimensional region specified by ivolume and the two-dimensional boundaries specified in other commands are included in energy calculations. Total, instantaneous energy is calculated for each component in the electric and magnetic fields and then summed. These individual energies are accessible to the OBSERVE command by the argument, ELECTRIC or MAGNETIC, and arguments, 1, 2, or 3, representing the component. However, these energies are calculated only once every istep time steps, due to the expense of the calculation.

Electromagnetic energy can flow out of the simulation through boundaries defined by the VOLTAGE and LOOKBACK commands. The Poynting flux through each such boundary is integrated over time since the beginning of the simulation, resulting in the total energy lost

through each boundary. These energies are accessible to the OBSERVE command by arguments, VOLTAGE and LOOKBACK, and by integer arguments representing the boundary number.

At any given instant, the total energy of particles and fields must balance the energy gained by created particles plus the energy lost through boundaries or destroyed particles. In other words, the residual energy, defined as the sum of these energies, must be zero. Of course, numerical approximations and round off conspire to make it non-zero, so residual energy, relative to total energy, is a measure of simulation integrity. These quantities, total energy, residual energy, and the ratio of residual to total are accessible to the OBSERVE command by the arguments, TOTAL, RESIDUAL, and RATIO, respectively. Also, the rates of change of these and all other energies calculated are available through the OBSERVE command.

Restrictions:

The electromagnetic calculation region specified by the area indices must not include undefined regions of the spatial grid; otherwise, instantaneous electromagnetic energies will be invalid. Also, the area must include all regions into which waves may propagate for total simulation energy and residual energy to be valid. As a consequence, all two-dimensional boundaries through which waves propagate must coincide with the boundaries of the calculation region.

See Also:

OBSERVE

BEXTERNAL

Function: Specifies spatially independent static magnetic fields.

Format:

BEXTERNAL b1ext b2ext b3ext ;

Arguments:

b1ext - x1-component of magnetic field (W/m^2).
b2ext - x2-component of magnetic field (W/m^2).
b3ext - x3-component of magnetic field (W/m^2).

Description:

The basic usage of this command is to define external fields which are constant in time and space. Optionally, this command may be used to define fields which vary in space. The command is internally connected to optional functions "AEX1", "AEX2", "AEX3", corresponding to spatial coordinates 1 through 3. For any of these for which a function is defined in the input, the spatial dependence for each field is the product of these spatial dependencies. The functions affect the external field only if a BEXTERNAL command is present.

The BEXTERNAL command defines an applied magnetic field independent of the dynamic electromagnetic field solution. This is a static field which affects only the particle forces in the kinematics algorithm, in addition to the forces due to the dynamic fields. Any or all of the three components of a magnetic field may be applied.

Restrictions:

1. Time-domain only.

2. Used only in conjunction with particles.

See Also:

BEX-READ

Example:

```
BEXTERNAL 0 10 0. ;
FUNCTION "AEX2 (X1) = SIN (100.0 * 2.0 * 3.1416*X1)" ;
```

This pair of commands defines an external field along the X2 coordinate that is independent of X2 and X3 and which varies sinusoidally along X1.

Function: Specifies spatially varying static magnetic fields.

Format:

BEX-READ afile scale ;

Arguments:

afile - input file (alpha).
= arbitrary, user-defined.
scale - scaling factor for input (unitless).

Description:

The BEX-READ command takes values for the static magnetic field supplied by a file and interpolates these values to the simulation grid. The static fields are added to dynamic magnetic fields to be used only for computing particle kinematics.

The fields obtained from the file, afile, can be modified by the scaling factor, scale. Additional BEX-READ commands add their magnetic fields to the existing eternal field.

In cylindrical coordinates, the command currently works only if the applied field has azimuthal symmetry. The file contains up to three sets of magnetic field data describing the r, θ, and z components of the field as a function of z and r.

The file contains unformatted data. The first record contains five integers stored in binary:

IZ IR ITHETA NZ NR .

The first three integers indicate which components of the magnetic field are non-zero. A zero in any of those fields indicates that field is zero. The last two integers indicate how many field data points are upcoming in the file describing the field, with NZ the number of points along the Z direction and NR the number of points along the R direction.

The second record contains a list of the Z and R coordinates at which the magnetic field is defined in the file. The Z coordinates are entered first, followed by the R coordinates, all in monotonically increasing order.

Subsequent records describe each of the non-zero magnetic field components. Each record exists only if the associated field is non-zero as indicated in record one. For field components which are non-zero, one record exists containing IZ*IR elements listed in order of first increasing z, then increasing r.

Restrictions:

1. Time-domain only.
2. Used only in conjunction with particles.
3. No more than five BEX-READ commands may be specified.

Function: Puts a comment into input that does not echo on output.

Format:

C ... ;

Arguments:

... - represents any string of input with or without quotes. Quotes must be used if string includes the command delimiter character.

Description:

The C command is used to put comments into the input file that do not show up in the output file. It can be used to comment out commands, to add comments after commands on the same line, and for many other purposes.

The difference between the C command and the COMMENT command is that the C command does not echo in the output. The difference between the C command and the Z command is that the Z command writes the message "Command Ignored" in the output file.

See Also:

COMMENT
Z

Examples:

```
C This sample input shows uses of the C command. ;
C A comment with the C command can split across
        lines. ;
DEFINE X 10 ;
C A comment here can say why X is being set to 10. ;
C The following C command comments cut a TITLE
        command. ;
C TITLE "simulation title" ;
```

CABLE

Function: Specifies coaxial cable drive for electromagnetic excitation of simulation space.

Format:

CABLE atype rsize acurrent(t) adir iline ;

Arguments:

atype	-	cable type (alpha). = STUB - coaxial E-probe. = LOOP - coaxial H-probe.
rsize	-	length of stub (m) or area of loop (m^2).
acurrent(t)	-	temporal function name (alpha).
adir	-	alignment and axis (integer). = X1ALIGN, X2ALIGN, X3ALIGN = X1ANTI-ALIGN, X2ANTI-ALIGN, or X3ANTI-ALIGN
iline	-	region containing stubs (6 integers).

Description:

This command allows the specification of a field driver in a single cell of the simulation. Within that cell the user can specify the length or area of the driver and the current in the driver as a function of time. The field argument adir specifies the direction of E at the probe for a coaxial E-probe (parallel to the wire) and is the direction of H at the probe for a coaxial H-probe (normal to the surface of the loop).

The driver is placed by SOS in the specified cell at the location at which the field that is driven is defined. Care must be taken near boundaries not to put the driver inside in a conductor.

Restrictions:

1. No more than ten CABLE commands may be specified.
2. Time-domain only.
3. Application region is currently restricted to a single cell.

Function: Reads and processes a command file.

Format:

CALL acall ;

Arguments:

acall - file name (alpha).

Description:

The CALL command reads and executes a command file. Nested CALL commands are allowed.

Restrictions:

1. CALL commands cannot be nested more than five levels deep.

2. CALLed files must be terminated with a RETURN command.

See Also:

RETURN

Examples:

In developing a template for analysis of a generic device, one might wish to isolate the actual input data for a specific design from the generic template commands. Thus, at the point in the template where design data is required, the command ,

```
CALL DESIGN.DAT ;
```

could be inserted. This file, design.dat, would then contain the design-specific data. It would consist of a series of commands to be processed in order at the point where the initial call is made. The same syntax rules apply to commands in the CALLed file as if they occurred serially in the CALLing file. However, the CALLed file (design.dat, in this example) must be terminated with a RETURN command.

CAPACITOR

Function: Specifies capacitive gap location and properties.

Format:

CAPACITOR capacitance axis isurface ;

Arguments:

capacitance	-	capacitance per unit length (f/m).
axis	-	capacitance axis (integer). = X1, x1 axis. = X2, x2 axis. = X3, x3 axis.
isurface	-	surface in terms of grid cells region (6 integers) in which capacitance applies. Surface must be exactly one cell wide on capacitance axis and one or more cells long.

Description:

This command enables the inclusion in the simulation of capacitors and capacitive gaps too small to be defined with cells in the grid. A subgrid model is employed to create the effect of a capacitor of specified strength in the specified cells. This model is valid both for time-domain and frequency-domain.

To define a capacitor gap, first define a cell-wide gap as shown in the following figure. The model assumes that the gap occurs between one-cell thick sheets, representing perfect conductors. Examples illustrated include an open-ended gap, a close-ended gap, and application at a corner. The physical gap will normally be substantially smaller than the cell used to represent it.

The arrows shown in the figure indicate the locations of electric fields which would be affected in these three examples. The arrows also indicate the orientation of the capacitive effect (along x_3 in all three examples). The argument axis, is simply the coordinate axis of the altered electric fields (3 in all three examples). The surface identifies the spatial location. The initial and final indices must be equal for one of the coordinates (x_1 in the examples). The indices must differ by unity in one of the other coordinates (x_3 in the examples). Thus, the spatial specification is equivalent to the plane having a height of one cell.

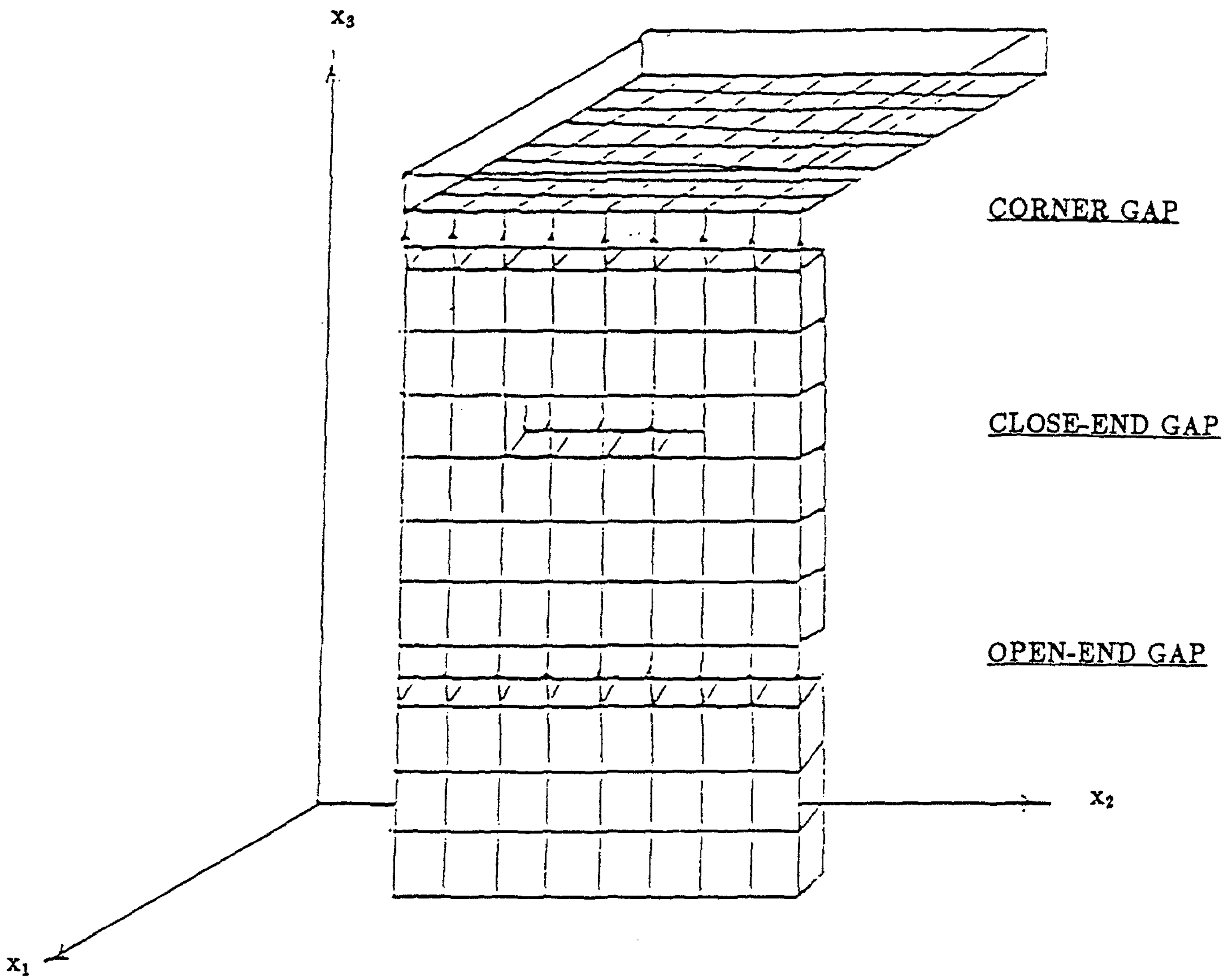


Figure 5.1. Specification of Input Data for Capacitive Gap.

CAPACITOR

The capacitance may be set to any value larger than the natural capacitance of the grid cell, but in general it will be set to a value that is equivalent to having a gap that is less than one cell thick. For example, in the simple gap shown in the following figure, the capacitance is

$$C = \varepsilon_o \frac{w}{d} .$$

Restrictions:

1. No more than ten CAPACITOR commands may be specified.

See Also:

STRUCTURE

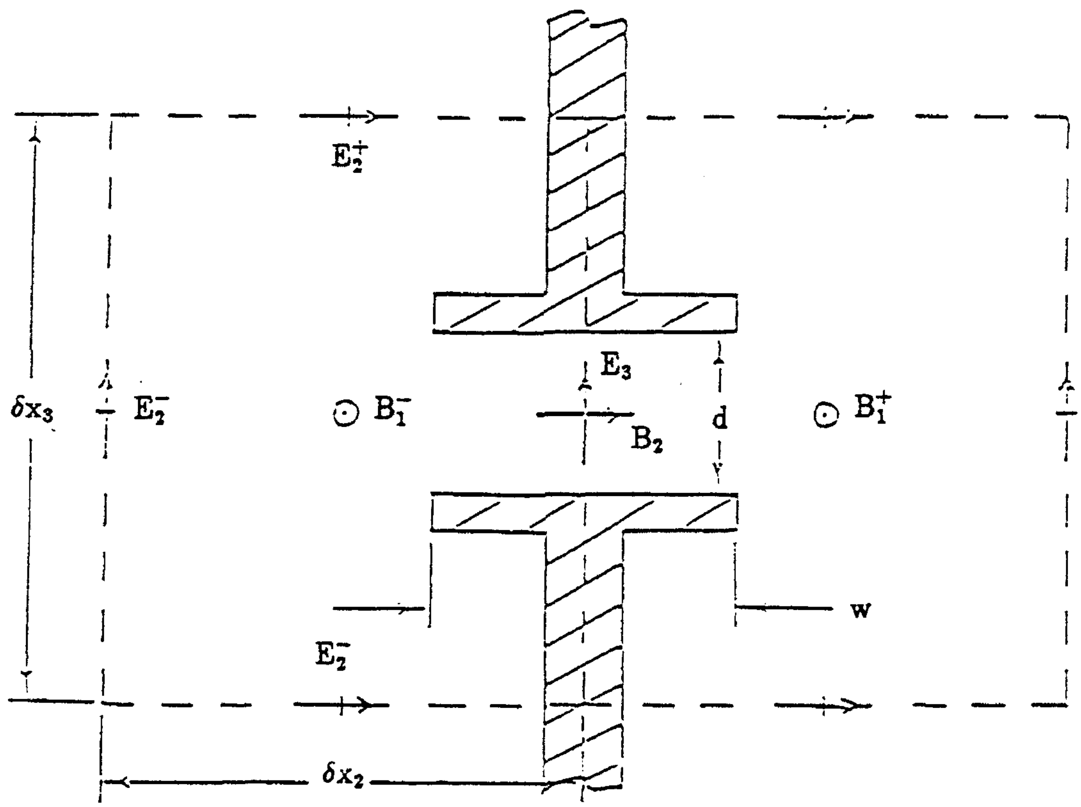


Figure 5.2. **Cross Section of a Capacitive Gap Aligned With X3.**

CHEMISTRY

Function: Specifies treatment of ionized air.

Format:

CHEMISTRY idtime aspeopt rair fwater asource eele rele egam rgam
eion ivolume ;

Arguments:

idtime	- time step interval (integer).
aspeopt	- species integration option (alpha). = EXPLICIT. = EXPONENTIAL.
rair	- air density ratio (unitless).
fwater	- water vapor fraction (unitless).
asource	- photon source label (alpha). = arbitrary, user-defined.
eele	- compton electron energy (Mev).
rele	- compton electron range (m).
egam	- incident photon energy (Mev).
rgam	- photon mean free path (m).
eion	- ion pair creation energy (ev).
ivolume	- volume of simulation to apply air chemistry.

Description:

The CHEMISTRY command specifies a ground-burst source EMP model of air conductivity with three species: electrons, negative ions, and positive ions. The electron avalanching and attachment rate expressions and species mobilities are calculated as a function of pressure, water vapor, and electric field and, with the resulting ionized species densities, are used to calculate the conductivity, which in turn is used to update the Compton current and the electric field. This calculation is performed every idtime steps. The differential equations governing species number densities can be integrated by either an explicit-centered-difference or an implicit-exponential-difference algorithm, chosen by the appropriate value of aspeopt. The ratio of air pressure to one atm is given by the unitless parameter, rair, and the water vapor fraction is given by fwater. The photon source geometry and time history, eele, is the average energy of the Compton electrons generated by gamma rays of energy, egam, and rele is the mean forward range of these Compton electrons. Eion is the average energy expended for the creation of one ion-pair in the gas being used. The portion of the three-dimensional grid containing air is specified by ivolume.

Reference:

B. Goplen, R. E. Clark, and J. McDonald, "An Air Chemistry Algorithm for SOS," MRC Technical Report, MRC/WDC-R-043, September 1983.

Restrictions:

1. Time-domain only.

2. This model is valid only at pressures greater than 0.05 atm.

See Also:

PHOTON
SOURCE

CLIGHT

Function: Alters the free-space permittivity and permeability.

Format:

CLIGHT feps0 fmu0 ;

Arguments:

feps0 - multiplicative factor applied to the free-space permittivity (unitless).
fmu0 - multiplicative factor applied to the free-space permeability (unitless).

Description:

CLIGHT permits the redefinition of the free-space constants so that the speed of light may be slowed while still preserving the correct electrostatic or magnetostatic limits. The free-space permittivity, ε_0, is replaced with feps0 $\times$ ε_0; the free-space permeability, μ_0, is replaced with fmu0 $\times$ μ_0. All occurrences of the free-space light speed, c, are replaced with $c/\sqrt{\text{feps0} \times \text{fmu0}}$. In particular, relativistic particle motion and the Courant condition are affected.

This feature is intended to provide a relaxed Courant condition (larger time step) and hence faster running simulations for low-energy, dominantly-electrostatic simulations. This is accomplished by setting feps0 to unity and fmu0 > 1, which simultaneously preserves the electrostatic limit, $\nabla \cdot E = \rho/\varepsilon_0$, while slowing the speed of light. The larger permeability constant results in a larger magnetic component, so that a physical criterion limiting the value of fmu0 is that the ratio of v×B to E in the Lorentz force remains small. Use of this option is particularly warranted in simulauions requiring resolution of ion time scales, because it offers a cost-effective alternative to the use of reduced ion mass.

Restrictions:

Both feps0 and fmu0 must be greater than zero.

Function: Places a comment into both input and output files.

Format:

COMMENT "comment" ;

Arguments:

comment - character string.
= arbitrary, user-defined.

Description:

The COMMENT command is used both to put comments into the input file and to have the comments echoed in the output file. The presence or absence of a COMMENT command will not affect the behavior of the simulation.

The use of any character except a (" ") is permitted within the comment. For example, a semicolon is permitted within the comment. The double quotes (" ") bracketing the comment are required.

The difference between the C command and the COMMENT command is that the C command does not echo in the output. The difference between the C command and the Z command is that the Z command writes the message "Command Ignored" in the output file.

See Also:

C
Z

CONVERT

Function: Assigns a length or angle in user-specified units to a variable.

Format:

CONVERT avar rnum aunits ;

Arguments:

avar	-	variable name must begin with B-H or O-Z. = arbitrary, user-defined.
rnum	-	length to be converted to meters (integer or real).
aunits	-	units in which rnum is given (alpha). = FEET, INCHES, MM, METERS, CM, RADIAN, or DEGREE.

Description:

This command takes a number (real or integer) in the units specified by the user and converts it to meters or radians. The resulting number is, in general, real rather than integer. This number is then assigned to the variable avar (which must begin with the letters (B-H, 0-Z) to correspond to a real variable). Once defined, this variable can be used in place of a number anywhere in any input command in which a numerical value is expected.

Examples:

The CONVERT command can be used to convert the distance 0.1 inches into .00254 meters and to assign the value .00254 to the variable CELL. The variable CELL can then be used in the X1GRID command to specify the cell size.

```
CONVERT CELL .1 INCHES  ;
X1GRID UNIFORM 10 2 0.0 CELL ;
```

Function: Performs a stability check on the electromagnetic field algorithm.

Format:

COURANT SEARCH ;
COURANT ipoint ;

Arguments:

ipoint - point indices (integers).

Description:

The COURANT command causes a calculation of the Courant stability criterion to obtain the upper limit of the electromagnetic time step. The use of a time step beyond that limit will cause numerical instability and catastrophic failure in the finite-difference solution of the fields. The alphanumeric argument, SEARCH will cause the stability criterion to be calculated for the entire grid. Specification of point indices, ipoint, will limit the stability check to the cell associated with those indices. This option may be desirable when a search for the worst case would produce an irrelevant result, e.g., where the smallest cells exist in a conductor.

Results will be printed out only if the appropriate DIAGNOSE command is specified by the user. If a violation of the maximum time step is detected, an error warning message will be printed and the simulation will be terminated. Note that, in general, the Courant stability criterion varies according to which field algorithm has been selected for the simulation. For example, the maximum allowable time step will generally be larger in the case of the time-biased algorithm. The Courant stability criterion for the time-biased algorithm is given by

$$\frac{1}{c^2 \delta t^2} > (\alpha_2^2 - 4\alpha_1\alpha_3) \sum_{i=1}^{3} \frac{1}{(\delta s_i)^2} .$$

See Also:

DIAGNOSE
FIELDS

CURRENTS

Function: Selects options in the current density algorithm.

Format:

CURRENTS fract ;

Arguments:

fract - temporal fraction (unitless).

Description:

The CURRENTS command provides the ability to apply temporal weighting to the current densities to reduce particle noise. The temporal fraction represents the contribution from the present time step. With the argument, fract, set to 1.0, only the present time step contributes. With any other value, a filtered result from previous time steps is computed.

Restrictions:

1. Temporal weighting of current densities can only be used when local charge conservation is not an absolute requirement.

2. It is also incompatible with the presence of certain models which the user may have chosen, such as the time-biased field algorithm.

3. Time-domain only.

4. Particles only.

Function: Specifies membrane damper location and properties.

Format:

DAMPER resist isurface ;

Arguments:

resist	- membrane resistance (ohms/square).
isurface	- surface indices (six integers).

Description:

The DAMPER command sets up a thin membrane of electrically conductive material that damps out electric fields. Only the components of electric field that are parallel to the membrane surface are attenuated, such as fields produced by an electromagnetic wave passing through the membrane. The attenuation is governed by the resistivity of the membrane, specified as ohms per square, the mks units for two-dimensional resistivity. The location of the membrane is given by the area indices, isurface.

Restrictions:

1. Dampers must be conformal and two-dimensional.
2. Only ten DAMPER commands may be specified.
3. Time-domain only.

DATAFILE

Function: Specifies diagnostic output data files to be written for storage and post-processing.

Format:

DATAFILE AFIELD atimer [i1] [i2] [i3] ;
DATAFILE FIELDS atimer [i1] [i2] [i3] ;
DATAFILE PARTICLES atimer aspecies fwrite ivolume ;

Arguments:

atimer	- timer name (alpha).
aspecies	- type of particle species to be written (alpha). = ELECTRONS. = IONS. = ALL.
fwrite	- fraction of particles to be written (unitless).
i1	- initial and final x1 indices and skip interval.
i2	- initial and final x2 indices and skip interval.
i3	- initial and final x3 indices and skip interval.
ivolume	- portion of simulation to write.

Description:

The DATAFILE command controls output of raw field and particle data to the files created by the DUMP command. Use of the DATAFILE command causes implicit execution of DUMP commands to turn on dumping of the data so there is no need to use the "DUMP TYPE FIELD" or "DUMP TYPE PARTICLES." The timer controls on which steps the data is output.

For FIELDS output, the values for all fields for each i3 plane are output, then the next i3 plane are output, then the next i3 plane, etc. Each field is output in order of minimum x,y to maximum x,y, varying x most rapidly. The skip interval is used to select output of field values for every n^{th} grid point on the grid cell.

When option AFIELD is used, interpolated field values at the full-grid point locations are output. When the option, FIELDS, is used the well-centered fields are output for the grid cells specified; thus, for the FIELDS option, the values output may correspond to locations outside the volume specified by an interpretation of [i1] - [i3] as grid points.

Restrictions:

1. At most two DATAFILE commands may be used - one for particles and one for fields (either option AFIELD or FIELDS).

2. When using DATAFILE to create an INITEB.DAT file (See FREQUENCY Command), use a skip interval of unity for each direction.

See Also:

DUMP

Function: Defines variables for use in other commands.

Format:

DEFINE aname value ;
DEFINE "aname = expression" ;

Arguments:

aname	-	name of variable being defined or redefined.
value	-	a numerical value (real or integer).
expression	-	arithmetic expression in the Fortran style composed of a combination of numbers, variables, intrinsic functions, and the operators add (+), subtract (-), multiply (*), divide (/), and exponentiate (**).

Description:

The DEFINE command assigns to the variable, aname, the value described for it in the command. Variable aname is assigned a real or integer value according to the Fortran convention based on the first letter in its name. If aname starts with I, J, K, L, M, or N, then the value is an integer. Otherwise, the value is real.

Once the variable, aname, is defined, it can be used in any other command in place of any numeric argument as a number. Note that once the variable, aname, is defined as a number, it will be used as such in all subsequent commands. Thus, if aname is the same as an ALPHA argument in any other command, aname will be converted to a number and will provide the wrong type of input argument to the command.

In the first format, the variable is ascribed the value, "value," and value should be an integer or real to correspond to the data type of aname. DEFINE will correctly convert between real and integer data types as necessary.

In the second format, the variable is ascribed the value that is the result of the evaluation of the arithmetic expression, "expression." The expression may include numbers, previously defined variables, basic Fortran operators, and any of the intrinsic functions listed in the following table. Most of the functions operate exactly as for Fortran, except that here MIN and MAX always expect exactly two arguments. The quotes shown above in the command format are required in the input.

The difference between the DEFINE and the FUNCTION commands is that the former always computes numerical values for the variable as the input data is being processed, while the latter computes numerical values during the actual simulation.

Restrictions:

1. The variable name, aname, must not be the same as an ALPHA argument used in any other command in the input file. For example, the variable name DATA cannot be used if the DATA option in the FUNCTION command is used.

2. Variables must be defined before they are used.

See Also:

FUNCTION

Examples:

The following sequence of commands computes the relativistic momentum of a particle.

```
DEFINE C 2.993E8 ;
DEFINE V 1.E6 ;
DEFINE RM 1.6E-19 ;
DEFINE GAMMA = (1 - (V/C)**2) ** (-.5)  ;
DEFINE P = GAMMA*RM*V ;
```

DELIMITER

Function: Specify command termination delimiter.

Format:

DELIMITER "achar" ;

Arguments:

achar - ascii character (alpha).

Description:

The DELIMITER command is used to change the MAGIC Command Language termination delimiter. The default value for the delimiter is a semicolon, ";". The user can change this to any legal ascii character or to a carriage return. If the delimiter is set to "RETURN", then the command line continuation over several lines is indicated with a dash "-". A long mathematical expression can be entered without continuation characters by enclosing the expression in double quotes. The delimiter takes effect on the first command following the delimiter command.

Restrictions:

1. The delimiter should not be set to a comma or period.

Examples:

The following examples illustrate the use of the delimiter command. The delimiter can be set to a slash "/" to allow use of older SOS input decks, written when the default delimiter was a slash.

```
DELIMITER "/" ;
C  The following commands have not changed. /
TITLE "TEST CASE 1" /
...
C  Reset the delimiter./
DELIMITER ";"  /
C  Enter the remainder of commands with the new
   delimiter ;
...
C  Reset the delimiter again ;
DELIMITER RETURN ;
C  This allows entering a command without using
- a delimiter; however, a dash must precede continuation
- lines
```

Function: Specifies diagnostic output.

Format:

DIAGNOSE aname interval istart istop ;

Arguments:

aname	- diagnostic name (alpha).
interval	- step interval (integer).
istart	- first step (integer).
istop	- last step (integer).

Description:

The DIAGNOSE command requests detailed intermediate output from various subroutines for diagnostic purposes. The desired output is selected with diagnostic label, aname. The diagnostics are generally given the names of subroutines in SOS. The printed output is provided at times specified by at the generalized steps specified by interval, istart, istop.

Example:

To examine all the full-grid and half-grid points right after they are calculated at the beginning of the simulation, one could use the command,

```
DIAGNOSE SPACING  1 0 0 ; .
```

To watch a frequency-domain calculation proceed, use

```
DIAGNOSE FREQUENCY 1 0 100 ;.
```

Function: Specifies grid-conformal dielectric blocks.

Format:

(with radiation induced currents)
DIELECTRIC arad asource eps conduct current ivolume ;
(no currents)
DIELECTRIC NO NULL eps 0.0 0.0 ivolume ;

Arguments:

arad	- radiation induced currents (alpha). = NO. = YES.
asource	- photon source name (alpha). = arbitrary, user-defined.
eps	- relative dielectric constant (eps/eps0).
conduct	- prompt conductivity constant ((mho/m)/(cal/cm^2/sec)).
current	- compton current constant ((A/m^2)/(cal/cm^2/sec)).
ivolume	- spatial volume (integers).

Description:

The DIELECTRIC command allows insertion of materials with dielectric and conductivity properties. The volume specified by grid indices, ivolume, is given the relative dielectric, eps. If ivolume overlaps a conductor, the conductor takes precedence. If two dielectrics overlap, the result is ambiguous unless an INTERFACE command is also used.

Prompt conductivity is added as

$$\sigma = \text{conduct} \cdot \dot{\gamma},$$

where $\dot{\gamma}$ is the dose defined as the PHOTON command for ASOURCE.

Compton current is added as

$$J_c = \text{current} \cdot \dot{\gamma}.$$

Restrictions:

1. For frequency-domain, use arad = NO, asource = NULL, conduct = 0.0, and current = 0.0.

2. For conduct ≠ 0, use EXPDIFF or LINEAR algorithm in FIELDS command.

3. No more than ten DIELECTRIC commands may be specified.

See Also:

INTERFACE
FIELDS
PHOTON

DIPOLE

Function: Specifies a dipole drive surface area.

Format:

DIPOLE asource adir isurface ;

Arguments:

asource	-	photon source name (alpha). arbitrary, user-defined.
adir	-	dipole alignment and axis (integer). = X1ALIGN = X2ALIGN = X3ALIGN = X1ANTI-ALIGN = X2ANTI-ALIGN = X3ANTI-ALIGN
isurface	-	spatial surface (integers).

Description:

The DIPOLE command creates an electric dipole drive to provide a source of electromagnetic waves. The dipole drive models the electric fields produced when high-energy photons hit a conducting surface; a negatively charged sheet of electrons moves a small distance away from the surface while a positive charge remaining on the surface prevents the electrons from escaping. Because the electron distance is smaller than the spatial grid cell width, and simulating particles is expensive, the dipole drive is implemented as a subgrid model.

The source of photons used to generate the dipole field is indicated by the name, asource. This name refers to a separate PHOTON command which specifies the magnitude, location, temporal behavior, and wavefront shape of the photon source. The resulting dipole moment per unit is given by the formula

$$\partial_t Q_1 \;(\tau,r) = s(\tau)\,\chi\,(r),$$

where $s\,(\tau)$ is the temporal function from the PHOTON command, τ is the retarded time, $\chi\,(r)$ is the spatial function from the PHOTON command, and r is the distance from the photon source location.

The direction of the dipole, adir, specifies the spatial grid axis parallel to the dipole axis. The alignment indicates the negative pole of the dipole; it should match the surface normal of a co-located conducting surface. The location of the dipole surface is given by the volume

indices, isurface. Of these six, two matching indices should be the same, indicating an infinitesimal width surface.

Restrictions:

1. No more than ten DIPOLE commands may be specified.
2. Time-domain only.

See Also:

PHOTON

Function: Displays spatial grid and structure.

Format:

DISPLAY atype iaxis iplane ;

Arguments:

atype - display option label (alpha).
= GRID, plots two-dimensional grid lines at full-grid coordinate values, plus symmetry planes and free-space boundaries.
= MODEL, plots a cross section of structure, plus the presence of dipole and emission.

iaxis - viewing axis (integer).
= 1, normal to x2-x3 plane.
= 2, normal to x1-x3 plane.
= 3, normal to x1-x2 plane.

iplane - axis coordinate index (integer).

Description:

The DISPLAY command provides cross-sectional plots of the grid or model definition of the simulation. The cross section is perpendicular to the axis, iaxis. For "grid plots," the plot is of the defined grid and is thus independent of iplane. For "model" plots, the structure is shown without the grid; thus, the output depends on iplane. In model plots, symmetry planes are shown as dotted lines.

Example:

See Chapter 6.

Function: Enables loop over commands in the input file.

Format:

DO avar start stop step ;

Arguments:

avar - variable to loop over (alpha).
start - number specifying first value of avar (integer or real).
stop - last value of avar (integer or real).
step - step size for the avar (integer or real).

Description:

The DO command allows the user to repeat over a blocked set of instructions. The commands that are looped over are those between the DO and the ENDDO commands. The variable, avar, is the counter variable, and start, stop, and step are the initial, final, and increment parameters. The first time through the loop, avar is equal to start. Each time the loop is executed, step is added to avar. The block is repeated until avar is greater than stop. The parameters may be integers or reals. The number step must be positive. If start and stop are equal, then the loop is executed once. If start is greater than stop, the loop is not executed.

DO commands can be nested. A DO loop can lie within another DO loop as long as the range of the inner loop lies completely within the range of the outer DO loop. Each loop must have a different counter, and each loop must have a corresponding ENDDO.

Restrictions:

1. The number step must be positive.

2. If start, stop, and step are integers, avar should correspond to an integer variable name (I-N). The name avar should not be the same as an alpha argument used in the input file.

3. DO loop nesting is limited to a depth of four.

See Also:

ENDDO

Example:

```
DO K 1 3 1 ;
DEFINE x = x + 3;
STRUCTURE SPHERE BALLS SOLID CARTESIAN 1 1 x .3 ;
ENDDO ;
```

This example draws three spheres of radius .3 m a distance of 3 m apart.

Function: Specifies output for post-processing.

Format:

DUMP NAME afile-name-suffix ;
DUMP FORMAT aformat ;
DUMP TYPE atype [NOPLOT] ;

Arguments:

afile-name-suffix	-	ascii to append to GRD, FLD, or PAR to create file names (alpha).
aformat	-	format for dumping data (alpha). = ASCII. = BINARY.
atype	-	type of data to dump (alpha). = all geometric specifications. = data from contour plots. = MATRIX. = FIELDS. = GRID, grid definition. = PARTICLES. = RANGE, data from range plots. = VECTOR, data from vector plots. = VIEW, data from view plots.

Defaults:

Keyword	Argument	Default Value
NAME FORMAT	file-name-suffix aformat	FILE BINARY

Description:

This command enables the dumping of data in a format that can be read by POSTER for graphical post-processing and by SOS for other simulations. The data is sorted and dumped to three files depending on the type of data. A particle file named PAR 'afile-name-suffix'.DAT receives data about particles, a field file named FLD 'afile-name-suffix'.DAT receives data about fields, and a grid file named GRD 'afile-name-suffix'.DAT receives any other data. It is not important for the user to know what data goes where, since other software reading the files will

know which file contains each type of data. Three files are created when the simulation begins, but any of the files for which no data is dumped is deleted at the end of the simulation.

The user has some control over the format of the files via aformat. Setting aformat to ASCII causes the files to be output in ASCII, which is useful for transferring data between computers. Setting aformat to BINARY causes the files to be output in binary, which is compact for saving disk space. There is no way to control individually the format for the three files: GRD, FLD, and PAR.

The basic way DUMP works is to intercept data as it is about to be plotted. This is true for the dump types: VECTOR, RANGE, and VIEW. Thus, for these dump types, data is dumped based on the plots selected in the commands of those names. VECTOR causes the two fields to be dumped for the locations at which the vectors in the plots are to be drawn. RANGE causes the set of (x,y) values of the curve to be dumped. VIEW causes a compressed representation of the portion of the structure requested for viewing to be dumped. The option NOPLOT causes the plots not to be plotted so that the data is dumped only.

Other dump types do not connect to plotting commands. These are GRID, MATRIX, FIELDS, and PARTICLES. MATRIX causes the matrix containing the bit map of the geometry to be output. FIELDS and PARTICLES couple to the DATAFILE command. Turning on FIELDS and PARTICLES still requires use of the DATAFILE command to set the step and volume controls for output.

See Also:

RANGE
VECTOR
DATAFILES
VIEW

Examples:

The following set of DUMP commands will cause files to have the names GRD201.DAT, FLD201.DAT, and PAR201.DAT, which is useful if simulations are given unique numbers. The data will be in binary, and the data requested by RANGE and VECTOR commands will be dumped. Also, VECTOR plots will not be created.

```
DUMP NAME "201" ;
DUMP FORMAT BINARY ;
DUMP TYPE RANGE ;
DUMP TYPE VECTOR NOPLOT ;
```

Function: Turns on echo of processed commands to output file.

Format:

ECHO ;

Arguments:

None

Description:

The ECHO command enables output of processed commands to the output file. This command is used in conjunction with the NOECHO command.

See Also:

NOECHO

Examples:

```
C This input demonstrates the use of ECHO and NOECHO.
  In the process of defining ten metallic vanes, four
  coordinate locations must be shifted, and NOECHO and
  ECHO are used to suppress output from those commands
  as unnecessary.  For the commands between the NOECHO
  and the ECHO, output of the processed command will
  only occur if there is an error in the command. ;

DO I 1,10,1 ;
  NOECHO ;                                C Turn off output.    ;
  DEFINE "IX1 =  2 + (I-1)*NXPV" ;        C Output suppressed. ;
  DEFINE "IX2 =  5 + (I-1)*NXPV" ;        C Output suppressed. ;
  DEFINE "IX3 =  9 + (I-1)*NXPV" ;        C Output suppressed. ;
  DEFINE "IX4 = 12 + (I-1)*NXPV" ;        C Output suppressed. ;
  ECHO ;                                  C Turn on output.    ;
  CONDUCTOR VANE'I' ANTI-ALIGN     IX1,NGRDY     IX2,NGRDY
                                   IX2,IVANEY    IX3,IVANEY
                                   IX3,NGRDY     IX4,NGRDY   ;
ENDDO ;
```

ELSE

Function: Initiates conditional or unconditional branch execution.

Format:

```
ELSE ;
ELSEIF ;
ELSE IF ;
```

Arguments:

None.

Description:

See IF.

See Also:

IF
ENDIF

Examples:

This example allows either the TM mode or both TM and TE modes to be selected in the field algorithm using the variable, LFLAG.

```
DEFINE LFALG = 2 ;
DEFINE KMAX = 100 ;
DEFINE DTIME = 1.0E-9 ;
IF (LFALG.EQ.1) THEN ;
   FIELDS TM BIASED KMAX DTIME ;
ELSE IF (LFALG.EQ.2) THEN ;
   FIELDS ALL BIASED KMAX DTIME ;
ENDIF ;
```

Function: Specifies parameters and surfaces for field, photo, and beam emission.

Format:

EMISSION astruct atype aspecies charge rmass
asource aene(w) ncreat idtime anorm atran amomen
gamv xnorm ethresh pyield ephoton ftherm
cos1 cos2 cos3 ivolume ;

Arguments:

astruct - name of structure to emit from (alpha).
= arbitrary, user-defined.

atype - emission type (alpha).
= FIELD.
= PHOTO.
= BEAM.

aspecies - type of particle to emit (alpha).
= ELECTRONS.
= IONS.

charge - ion charge state, z (unitless).
= 0, for electrons.
> 0, for ions.

rmass - ion mass number, A (unitless).
= 0, for electrons.
> 0, for ions.

asource - photon source name (alpha).
= arbitrary, user-defined in PHOTON command.

aene(w) - energy function name (alpha).
= arbitrary, user-defined.

ncreat - creation number density (particles/cell).

idtime - time step interval (integer).

anorm - normal coordinate option (alpha).
= RANDOM, random spacing over xnorm.
= FIXED, creation at xnorm.

atran - transverse coordinate option (alpha).
= RANDOM, random between full-grid points.
= FIXED, fixed at half-grid points.

amomen - momentum component option (alpha).
= GAMV, normal momentum at gamv.
= PHOTO, uncorrelated photoemission.
= THERMAL, transverse thermalization.

		= COSINES, directional cosines.
gamv	-	gamma times velocity (m/sec).
xnorm	-	maximum normal distance (m).
ethresh	-	field emission threshold (V/m).
pyield	-	photoemission yield (electrons/photon).
ephoton	-	average photon energy (keV).
ftherm	-	thermalization fraction (unitless).
cos1	-	p1 directional cosine (unitless).
cos2	-	p2 directional cosine (unitless).
cos3	-	p3 directional cosine (unitless).
ivolume	-	specification of portion of structure to emit from.

Description:

The EMISSION command designates the surface of a conductor as a source of charged particles and specifies how those particles are to be emitted from that surface. Particle flow can be used to model electric field breakdown on a conductor, photoemission of electrons due to a high-energy photon source, or a charged particle beam. The charged particles produce current densities which give rise to electromagnetic waves and self-consistently influence the particle motion.

The emission type, atype, allows three emission process options: FIELD, PHOTO, and BEAM. FIELD selects emission by electric field breakdown; once a given threshold is exceeded, charge is produced in response to the electric field strength at the emission surface. PHOTO selects emission by the photoelectric effect; charge is produced in response to the amount of photon energy incident on the emission surface. BEAM selects emission by the creation of a specified beam.

For each type of emission, certain of the command arguments are important, and some of them are ignored.

For FIELD emission, charge is emitted to neutralize the electric field at the emission surface. A threshold parameter, ethresh, allows a specification that the magnitude of the electric field attain a certain value before causing breakdown. Once breakdown occurs, the charge emitted is that required to cause the electric field at the surface to vanish. For field emission, amomem=PHOTO should not be used. Field emission does not use the following inputs: aene(w), asource, pyield, and ephoton.

For BEAM emission, charge is emitted to create a current density (A/m^2), specified by

$$j = \text{atem}(\text{time} + \text{tadvance}/c) \cdot \text{aspace}(\text{dist}) ,$$

where the functions are defined in a PHOTON command. Here, atem is the temporal function, tadvance is the retarded time advancement, aspace is the spatial function, c is the speed of light, and dist is the distance from the location of the photon source to the point of emission. Note that atem does not include a factor for the distance from the photon source to the point of emission. The use of a photon source is simply a way to allow input of general beam specifications. For beam emission, amomem=PHOTO should not be used. Beam emission does not use the following inputs: aene(w), pyield, ephoton, ethresh.

For PHOTO emission, charge is emitted to create a current density (A/m^2) given by

$$j = -41.87 \text{ atem(time+tadvance/c-dist/c)} \cdot \text{aspace(dist)} \cdot \text{(pyield/ephoton)},$$

where the meaning of the variables is the same as for BEAM emission above. For photo-emission, amomem=PHOTO should be used, with aene(w) to specify the distribution of charge vs energy. Photoemission does not use the following input: gamv, ethresh, ftherm, cos1, cos2, cos3.

The emission of electrons or ions is specified by aspecies. If aspecies=ELECTRONS, then the charge and rmass of the electrons is known internally to SOS, so the two parameters, charge and rmass, may be set at zero. If aspecies=IONS, then any positive-charge, positive-mass ion may be specified using the parameters, charge and rmass. The parameter, rmass, expects input in atomic mass numbers, so the mass of the proton is approximately unity.

Other controls available include the frequency of emission (idtime), the number of particles to create per cell, the position of the particles emitted, and their momentum. The distance of emission from the surface of the structure, xnorm, should be such that the particle is in a cell adjacent to the emission structure. Momentum for field and beam emission may be specified as entirely normal to the emission surface (amomem=GAMV), as thermalized (amomem=THERMAL), or at a specific angle relative to the simulation (amomem=COSINES). For amomem=GAMV, the quantity gama=gammav is input and the quantities, ftherm, cos1, cos2, and cos3, are ignored. For amomem=THERMAL, gamv and the quantity 0<=ftherm<1 are input and the quantities, cos1, cos2, and cos3, are ignored. For amomem=COSINES, values for gamv, cos1, cos2, and cos3, are input, and ftherm is ignored. The values of cos1, cos2, and cos3 are the cosines between the desired angle of emission and the X1, X2, and X3 axes, respectively.

Restrictions:

1. Time-domain only.

2. Particles only.

3. No more than ten EMISSION commands may be specified.

See Also:

STRUCTURE
PHOTON
FUNCTION

Function: Enables loop over commands in the input file.

Format:

ENDDO ;

Arguments:

None.

Description:

The DO command allows the user to repeat over a blocked set of instructions. The commands that are looped over are those between the DO and the ENDDO commands. DO commands can be nested. A DO loop can lie within another DO loop as long as the range of the inner loop lies completely within the range of the outer DO loop. Each loop must have a different counter and each loop must have a corresponding ENDDO.

See Also:

DO

Example:

```
TIMER RTIMER PERIODIC 250 1000 250 ;
DO ICELL 10 50 10 ;
RANGE RTIMER
        1 0 0  E1
        ICELL 2 30 ICELL 50 30 ;
ENDDO ;
```

This example causes the output of five range plots at time step 900, one each for five different slices through X2 taken at five different values of X1 (X1 = 10, 20, 30, 40, 50).

ENDIF

Function: Terminates conditional execution logic.

Format:

ENDIF ;

Arguments:

None.

Description:

See IF.

See Also:

IF
ELSE

Examples:

This example allows either the TM mode or both TM and TE modes to be selected in the field algorithm using the variable, LFALG.

```
DEFINE LFALG = 2 ;
DEFINE KMAX = 100 ;
DEFINE DTIME = 1.0E-9 ;
IF (LFALG.EQ.1) THEN ;
        FIELDS TM BIASED KMAX DTIME ;
ELSE IF (LFALG.EQ.2) THEN ;
        FIELDS ALL BIASED KMAX DTIME ;
ENDIF ;
```

Function: Converts coordinate to grid index.

Format:

ENGRID aaxis avar rval [aopt] ;

Arguments:

aaxis	-	coordinate axis (alpha). = X1, X2, or X3.
avar	-	name of integer variable to store result in (alpha).
rval	-	coordinate value (m or rad).
aopt	-	index option (alpha). = CELL, returns cell index that rval is in (default). = FULL, returns full-grid index nearest to rval. = HALF, returns half-grid index nearest to rval.

Description:

The ENGRID command returns a grid index based on rval and aopt. If aopt is not specified, CELL is assumed. For the CELL option, ENGRID returns the index of the cell in which rval is located, and that index value is stored in avar. The integer variable, avar, must be a legitimate integer variable. For the FULL option, ENGRID returns the index of the full-grid point nearest to rval, and for the HALF option, ENGRID returns the value of the half-grid point nearest to rval. For non-uniform grids, the determination of nearest grid point includes the implied stretching of the coordinate system. That is, it depends on which half- or full-cell the rval lies within, rather than which grid position is physically closer.

Restrictions:

The axis must be defined with an XnGRID command before the ENGRID command is used. If rval lies outside of the defined grid, the lowest or highest defined index is given, depending on whether rval is below or above the defined grid, respectively.

See Also:

SNAPGRID
UNGRID
XnGRID

Examples:

1. The following command finds out what cell on the X1-axis that 3.71E-3 is located in and the index of that cell is placed in the integer variable, ICELL.

```
X1GRID UNIFORM 100 2 0.0 1.4E-4 ;
ENGRID X1 ICELL 3.71E-3 CELL ;
```

2. The following command is the same as that above, except that CELL is not specified. The result is the same because CELL is the default.

```
ENGRID X1 ICELL 3.71E-3 ;
```

3. The following command finds the full-grid point on the X2-axis that 0.93 is nearest to and places the index of that point in IWINDOW. IWINDOW could later be used in a PLOT WINDOW command.

```
X2GRID UNIFORM 12 2 0.0 0.1 ;
ENGRID X2 IWINDOW 0.93 FULL ;
```

Function: Specifies electromagnetic field algorithm and parameters.

Format:

(centered-difference, linear-conductivity, and exponential-difference algorithms)
FIELDS algor ktime dtime ;

or

(time-biased algorithm)
FIELDS algor ktime dtime tcoef [,...] iters rcoef [,...] ;

Arguments:

algor	-	field algorithm (alpha). = CENTERED, centered-difference (explicit). = BIASED, time-biased. = LINEAR, linear conductivity (implicit). = EXPDIFF, exponential difference (implicit).
itime	-	maximum number of time steps (integer).
ktime	-	time step (sec).
tcoef	-	three temporal coefficients at t+3/2dt, t+1/2dt, and t-1/2dt (unitless).
iters	-	number of relaxation coefficients (integer).
rcoef	-	relaxation coefficients (unitless).

Description:

The FIELDS command specifies parameters for the electromagnetic field algorithm. There are four choices for the numerical algorithm. The centered-difference algorithm is the basic leap-frog technique. It is usually the best choice for simulations not involving particles.

The time-biased algorithm is an implicit scheme used to lessen the effects of numerical instabilities arising from particle noise. The basic idea involves iterating between electric and magnetic field calculations during a single time step. Monotonically decreasing relaxation coefficients are applied to corrections in the electric field solution. It is the most expensive of the four field algorithms; however, a somewhat larger time step may be used. The temporal coefficients, tcoef, represent fractional contributions from magnetic field differences at three points in time (t+3/2dt, t+1/2dt, and t-1/2dt). These coefficients are subject to the constraints,

$$\alpha_1 + \alpha_2 + \alpha_3 = 1$$

$$\alpha_1 > \alpha_3$$

$$\alpha_2^2 - 4\,\alpha_1\,\alpha_3 > 0.$$

The relaxation coefficients, rcoeff, are typically in the range zero to one, with the first coefficient always equal to one. Table 5.1 presents a variety of sets of temporal and relaxation coefficients, all of which assign $\alpha_3 = 0$.

Restrictions:

1. The time-biased algorithm must not be used in conjunction with a current density temporal weighting fraction less than one.

2. All field algorithms have an upper limit to the time step used.

See Also:

COURANT
CURRENTS

Table 5.1. Coefficients for the time-biased algorithm.

α_1	I	τ_i
0.125	2	1.0, 0.60494
	3	1.0, 0.75385, 0.60494
	4	1.0, 0.83944, 0.68410, 0.60494
0.25	2	1.0, 0.36000
	3	1.0, 0.52941, 0.36000
	4	1.0, 0.65759, 0.44305, 0.36000
0.50	3	1.0, 0.20000, 0.11111
	4	1.0, 0.29912, 0.15022, 0.11111
	5	1.0, 0.39559, 0.20000, 0.13383, 0.11111
	6	1.0, 0.48267, 0.25457, 0.16470, 0.12613, 0.11111
0.75	6	1.0, 0.13458, 0.05385, 0.03182, 0.02349, 0.02041
	7	1.0, 0.17381, 0.06984, 0.04000, 0.02803, 0.02260, 0.02041
	8	1.0, 0.21488, 0.08768, 0.04944, 0.03359, 0.02591, 0.02205, 0.02041
	9	1.0, 0.25676, 0.10712, 0.60001, 0.04000, 0.03000, 0.02459, 0.02169, 0.02041
	10	1.0, 0.29857, 0.12791, 0.07159, 0.04717, 0.03472, 0.02775, 0.02371, 0.02144, 0.02041
	11	1.0, 0.33964, 0.14980, 0.08410, 0.05504, 0.04000, 0.03142, 0.02624, 0.02308, 0.02125, 0.02041
	12	1.0, 0.37943, 0.17256, 0.09743, 0.06355, 0.04579, 0.03551, 0.02919, 0.02517, 0.02262, 0.02111, 0.02041
	16	1.0, 0.52021, 0.26799, 0.15728, 0.10308, 0.07336, 0.05556, 0.04418, 0.03654, 0.03125, 0.02750, 0.02481, 0.02291, 0.02161, 0.02080, 0.02041

FREESPACE

Function: Absorbs outgoing waves as they leave the simulation.

Format:

FREESPACE acond(x) adir ivolume ;

Arguments:

acond(x)	-	conductivity function name (alpha). = arbitrary, user-defined.
adir	-	alignment and axis (alpha). = X1ALIGN. = X2ALIGN. = X3ALIGN. = X1ANTI-ALIGN. = X2ANTI-ALIGN. = X3ANTI-ALIGN.
ivolume	-	volume indices (integers).

Description:

The FREESPACE command is used to simulate an outgoing wave escaping into the infinity of free space. In the physical world, such a wave will not return to the region being investigated. In a simulation, such a wave could reflect off the boundary of the simulation and re-enter the simulation. The FREESPACE command is used to absorb outgoing waves at the boundary of the simulation so that the reflection, if any, is small and not important.

The FREESPACE command is similar in purpose to the LOOKBACK command, except that it is more appropriate in some circumstances and less in others. The FREESPACE command works for a broad frequency band. Also, it is more stable numerically for long simulations. On the negative side, it uses more grid cells than the LOOKBACK command, and the conductivity function that it uses must be tuned to the individual simulation.

The conductivity function, acond(x), specifies the resistivity which is used to damp out electromagnetic waves. The resistivity (mhos/m) is a function of distance along the direction of wave propagation, measured from the inner edge of the application area.

The volume indices specify the volume in which the prescribed conductivity is applied. The specified volume must have width, height, and depth and the final indices must be greater than the initial indices. One or more edges of the specified area should be contiguous with the outer boundary of the simulation.

Minimization of reflection of the wave back into the simulation is achieved by adjusting the conductivity function and the free-space area in the direction of the propagation. Wide free-space areas with slowly rising absorption values provide the best results and use the most computer time.

Restrictions:

1. Time-domain only.

2. No more than six FREESPACE commands may be specified.

3. Each FREESPACE volume is limited to a maximum of twenty cells in the direction ot propagation.

See Also:

FUNCTION
LOOKBACK

Examples:

```
FREESPACE   ABSORB   X1ANTI-ALIGN   10   15   1 10 1 10 ;
FUNCTION    "ABSORB(X) = X**2" ;
```

In the FREESPACE command, ABSORB is the name of a function specifying acond(x), the absorption as a function of distance: adir, X1ANTI-ALIGN indicates that waves will be coming from lower values of x1. The x1 range for the free-space boundary is i1=10 to i1=15.

The FUNCTION command defines the function ABSORB. The form for the function is

$$\text{acond}(x) = x^2 \ .$$

The code will account internally for the "zero" location in the x1 coordinate, so that the function will have the value zero at one boundary and value unity at the other.

It should not be assumed from this example that this form or normalization is appropriate to obtain minimum reflection for any particular simulation.

FREQUENCY

Function: Specifies algorithm to find resonant frequencies and fields.

Format:

FREQUENCY nmode wmax accuracy iterations ;

Arguments:

nmode	-	number of modes sought. A value of nmode as high as ten has been achieved.
wmax	-	highest frequency sought for the resonances. The algorithm always seeks the lowest modes.
accuracy	-	fractional accuracy sought for highest resonance (< .005 required). The algorithm will find each lower resonance to at least this accuracy.
iterations	-	number of total iterations allowed for locating all modes. SOS terminates and outputs the results it has found if this limit is reached.

Description:

The FREQUENCY command directs SOS to find resonant electromagnetic modes for the simulation structure, beginning with the lowest frequency. After all resonances have been found, the algorithm outputs the found resonant frequencies to OUTFIL.DAT in order of lowest frequency to highest. Additional output may be obtained using the DIAGNOSE command with the FREQUENCY option.

When a mode is degenerate, the algorithm identifies it once and moves on to the next resonant frequency. The resulting mode structure will generally contain mixed modes; the degree to which near-degenerate modes are separated is still an open question. Using the DIAGNOSE FREQUENCY option, it is possible to obtain the level of contamination in the modes plus an estimate of the next mode. Comparison of this estimate against the next mode found will indicate whether a mode has been missed.

For efficiency, select nmode to match real needs. Each mode requested takes time to identify; additionally, the accuracy to which each of the lowest modes must be found increases as more and more of the higher modes are requested.

The selection of wmax also impacts the efficiency of the code execution. The code will generally work significantly better when wmax is chosen to be near, but above, the expected frequency of the highest mode sought.

Of nmodes and wmax, it is currently preferred that nmode actually cause the termination of the code execution. In order for wmax to terminate execution, the code must find the next resonance above wmax to know that the search for resonances is done. This takes time, and if the lowest frequency above wmax is far above wmax, then the code may take longer to find it and terminate than it took to find all the other resonances.

The selection of fractional accuracy enables the code to save time by terminating when the results found are accurate enough for the user's needs. The input accuracy value specifies the required accuracy for the highest resonance to be found. The algorithm, in order to achieve that accuracy for the highest resonance, must identify each lower resonance to at least that accuracy and typically to at least approximately twice that accuracy. Thus, to find two resonances with the second accurate to .001. the first resonance will be found to .0005. To find three resonances with the third accurate to .001, the algorithm will find the second to an accuracy of approximately .0005, and the first to approximately .00025, etc.

The FREQUENCY command is incompatible with the FIELDS command. The code's performance is sensitive to each computer's random number generator. The quality of the results will generally be the same, but the accuracy to which any particular resonance is found may vary, as will mode mixing in the case of degenerate modes. Similar effects may also appear if the problem specification is changed even slightly, even on a single computer. Each result will generally be as good as any other, just different within the accuracy specified by the accuracy values output for each frequency.

Restrictions:

The value of wmax must be less than half of the highest frequency mode for the discretized device as determined from the cell size. The code computes the value, and if wmax is greater than one-half of that value, the code resets wmax equal to one-half of the highest of the discretized device. This is for efficiency plus the numerically found frequencies above or even near one-half of the highest frequency of the discretized device. If those higher frequencies are desired, they can be obtained by reducing the cell size. That will increase the highest frequency of the discretized device and make the numerically found frequency values more accurate.

Function: Defines functions to be referenced by other commands.

Formats:

FUNCTION afn(a1,a2,...an) = expression ;
FUNCTION af1 DATA npair xval, yval [xval, yval,...] ;

Arguments:

afn	-	name of scalar function of up to 10 independent variables (alpha). = arbitrary, user-defined.
a1	-	first dummy argument (alpha). = arbitrary, user-defined.
a2	-	second dummy argument (alpha) . = arbitrary, user-defined.
an	-	n^{th} dummy argument (alpha). = arbitrary, user-defined.
expression	-	arithmetic expression in Fortran style and composed of a combination of dummy arguments, previously defined functions, intrinsic functions, variables, numbers, special values defined by other commands, and the operators, add (+), subtract (-), multiply (*), divide (/), and exponentiate (**).
af1	-	function name for a function of a single variable = arbitrary, user-defined (alpha).
npair	-	number of data pairs (integer).
xval	-	value of x for which corresponding yval=af1(xval) is given.
yval	-	af1(xval).

Description:

The FUNCTION command defines the form of a function which is then referenced in other commands requiring such functions. The meaning of the dummy arguments in the function depends upon the context of its use and can be different for the same function in different references. Usually the arguments are expected to be in units of seconds, meters, radians, or other MKS units. The dummy arguments must be enclosed in parentheses and separated by commas in the function definition. The function is referenced by its name; thus, each function must have a unique name.

Numerical data, a set of (xval,yval) data pairs, is entered in order of increasing x. For values of x between the values entered, y will be computed via linear interpolation from the

given values. For values outside the range, the nearest value is used. Extrapolation is not used. If the numerical data option is used to describe a multivariate function, then it is assumed that the data represents variation with respect to the first independent variable.

Arithmetic expressions are entered as strings in double quotes in Fortran style. The equal sign shown in the command format must be input. Any combination of the expression components previously listed for the expression may be used. The operators, add (+), subtract (-), multiply (*), divide (/), and exponentiate (**), have the same priority as in Fortran. Any of the intrinsic functions (see Table 5.2) may be included. Also, any function defined by a previous FUNCTION command may be included. The dummy arguments are entered to create the dependence of the result on the input values of those arguments.

In some of the codes in the SOS family, the FUNCTION command recognizes special values defined by other commands. For instance, in the SOS code, observers may be referenced in expressions in function commands. Observer values are referenced as OBSj, where the j represents the number that identifies the observer by its position in the input file. The value of OBSj when the value of the function is computed is the value of observer j as of the end of the previous time step. On time step 0, OBSj=0.

Restrictions:

1. The number of data pairs per function is limited to forty.

2. A function may not reference itself.

3. A function whose name duplicates an intrinsic function will replace the intrinsic function.

4. Function names must not duplicate variable names.

5. The use of double quotation marks to enclose the expression option is mandatory when the delimiter is a slash "/".

See Also:

DEFINE
OBSERVE

Examples:

1. The following commands create two functions. The first function, named SINP, is a sine wave of position and time. The second function, named SINWAVE, is a sine wave plus some noise.

```
DEFINE "C=2.998E8" ;
FUNCTION "SINP(X,T) = 10.*SIN(4E9*(X/C+T))"  ;
FUNCTION    "SINWAVE(X,T)    =    SINP(X,T)*    (1    +
        GAUSSIAN(0,.05))" ;
```

2. A command format contains the symbol, af (t, x). The desired input is linear in t and independent of x, e.g.,

```
FUNCTION PULSE DATA 2 0.0 0.0 1.0E-8 1,0 ;
```

The expression processor will interpret this data as a functional variation with the first independent variable (in this case, t). However, the data option cannot be used to describe variation with the second variable.

Table 5.2. Table of intrinsic functions.

Name	No. of Arguments	Function	Notes
SIN	1	Sine	1
COS	1	Cosine	1
TAN	1	Tangent	1
ASIN	1	Arc Sine	2
ACOS	1	Arc Cosine	2
ATAN	1	Arc Tangent	2
ATAN2	2	Arc Tangent	2,3
SINH	1	Hyperbolic Sine	
COSH	1	Hyperbolic Cosine	
TANH	1	Hyperbolic Tangent	
EXP	1	Exponential	
LOG	1	Natural Logarithm, ln	
LOG10	1	Common Logarithm, $\log_{10}$	
ABS	1	Absolute Value	
MIN	2	Minimum of (a_1,a_2)	4
MAX	2	Maximum of (a_1,a_2)	4
MOD	2	Remainder of a_1/a_2	
SIGN	1	Sign of a_1	5
RANDOM	0	Random number	6
GAUSSIAN	2	Gaussian random number	7
STEP	2	Step function	8
SQRT	1	Square root	

1 Arguments: are in radians.
2 Results are in radians.
3 First argument is sine of angle, second argument is cosine. The range for the result is $-\pi < \text{result} < \pi$.
4 Always two arguments -- different from Fortran.
5 If $a_1<0$, then result = -1; otherwise result = +1.
6 Result evenly distributed over range, $0 \leq \text{result} \leq 1$.
7 Result is Gaussian distributed: $\exp(-(\text{result}-a_1)^2/2a_2)$.
8 If $a_1>a_2$, then result=1; if $a_1=a_2$, then result = 1/2; otherwise, result=0.

Table 5.2. Table of intrinsic functions (continued).

Name	No. of Arguments	Function	Notes
BESSELJ0	1	Bessel function, J_0 (a1)	
BESSELJ1	1	Bessel function, J_1 (a1)	
BESSELJN	2	Bessel function, J_{a1} (a2)	1
BESSELJP	2	Bessel function derivative, J'_{a1} (a2)	
BESSELJNZ	2	Zeros of Bessel function, J_{a1} (a2)	2
BESSELJPZ	2	Zeros of Bessel derivative, J'_{a1} (a2)	2
BESSELYN	2	Bessel function, Y_{a1} (a2)	3
BESSELYP	2	Bessel function derivative, Y'_{a1} (a2)	3
BESSELYNZ	2	Zeros of Bessel function, Y_{a1} (a2)	2
BESSELYPZ	2	Zeros of Bessel derivative, Y'_{a1} (a2)	2

1 The first argument is the order of the Bessel function with a range of $0 \le a_1 \le 50$.
2 The second argument is the zero index with a range of $1 \le a_2 \le 10$.
3 The first argument is the order of the Bessel function with a range of $0 < a_2 < \infty$.

Function: Specifies output plots on system graphics or line-printer graphics.

Format:

GRAPHICS atype ;

Arguments:

atype - graphics output device (alpha).
= SYSTEM, system graphics.
= PRINTER, line-printer graphics.

Description:

The GRAPHICS command selects the means by which SOS produces its various plots. When agraph is specified as SYSTEM, SOS creates all of its graphical output using DISSPLA or NCAR (depending on how SOS is installed). These graphics packages produce a metafile containing all of the simulation plots. After the simulation, a postprocessor converts the metafile into images on a CRT, a pen plotter, or other selectable graphical output devices.

When agraph is specified as PRINTER, SOS creates its graphics using LPG, an SOS internal graphics package which formats plots for output on a line-printer. Lines in such plots are created using characters inside a block of 51 lines by 70 columns, so resolution is much less than that provided by the SYSTEM option. The line-printer plots are printed in the output file as they are produced.

Restrictions:

1. The DISSPLA and NCAR graphics packages are separate from SOS and therefore may not be available on a given computer system. In such a case, the only valid choice for agraph is PRINTER.

2. Certain commands in SOS will not produce line-printer graphics. Thus, if the PRINTER option is chosen, some of the desired plots may not be produced.

HALFGRID

Function: Provides an alternate halfgrid computation for radial spacing, improving the current at the axis.

Format:

HALFGRID ;

Arguments:

None

Description:

This command provides an option of setting the halfgrid spacing near r=0 for cylindrical coordinates. For certain simulations involving current near the axis, this alternate spacing avoids a mathematical current spike at r=0. This command is useful primarily when using particles.

Restrictions:

1. Not for use with cartesian coordinates.

2. Affect on frequency-domain calculation unknown.

Function: Provides conditional execution of commands.

Format:

IF (expression) [THEN] ;

Argument:

expression - logical expression to be evaluated (as per fortran)

Description:

The IF statement works together with ELSE (IF) and ENDIF statements to provide conditional execution. The THEN is optional for readability and compatibility with fortran.

See Also:

IF
ENDIF

Examples:

In the following example, the number of time steps, KMAX, is calculated from variables KCYC and NCYC, which have been entered previously in the input data. Here, KCYC represents the number of time steps in a RF cycle and NCYC represents the number of RF cycles.

```
DEFINE NCYC = 10 ;
DEFINE KCYC = 150 ;
!
IF (NCYC.EQ.0) THEN ;
  DEFINE KMAX = 1 ;
ELSE IF (NCYC.GT.0) THEN ;
  DEFINE KMAX = NCYC*KCYC ;
ENDIF ;
```

Notice that a negative value for NCYC will result in KMAX undefined, unless it is set elsewhere.

Function: Specifies boundary interfaces between regions of dielectric.

Format:

INTERFACE epsm epsp asnor isurface ;

Arguments:

epsm - relative dielectric constant in the region to the negative side of the interface (eps/eps0).

epsp - relative dielectric constant in the region to the positive side of the interface (eps/eps0).

asnor - axis normal to plane of interface (integer).
= X1, x1 axis.
= X2, x2 axis.
= X3, x3 axis.

isurface - surface indices (integers).

Description:

This command is used in conjunction with the DIELECTRIC command to treat correctly the boundary of dielectric regions. When two dielectrics have been defined which have a common surface, an interface command is used to identify the overlap. SOS does not locate the common surface automatically.

The interface command causes the overlap region to be assigned the average dielectric constant of the two dielectrics in the region, which is an approximate treatment of the boundary. The magnetic field at the boundary is discontinuous in this approximation, which is consistent with a similar approximation made for boundaries of dielectrics defined with the DIELECTRIC command.

Restrictions:

No more than ten INTERFACE commands may be specified.

See Also:

DIELECTRIC

Function: Controls journal facility.

Format:

JOURNAL aopt ["astring"] ;

Arguments:

aopt - option to turn journal on and off, and to review contents of journal. = ON, OFF, REWIND, or REVIEW.

astring - optional. Valid for REVIEW only. It selects lines from the journal with astring in it.

Description:

A journal file can be used to record all of the valid commands entered during an interactive session. They will be recorded in the order entered, including comments. Then the journal file may be edited to rapidly create a new input file. This is useful when the program is used interactively to create the initial input. The file is always created with the name JOURNAL.DAT. The ability to close and reopen the journal file will depend on the operating system. On the VAX, a new version of the file will be created.

Function: Specifies particle kinematics algorithm.

Format:

KINEMATICS kmultiple arelopt ;

Arguments:

kmultiple - partile-to-field time step ratio (integer).
= 1, [default].
> 1.

arelopt - relativistic kinematics option (alpha).
= NONRELATIVISTIC, without magnetic forces.
= RELATIVISTIC, with magnetic forces [default].

Description:

This command provides two controls to help reduce the cpu used for particle kinematics. The controls provide the most benefit when the speed of particles is significantly less than the speed of light. When this command is not used, the code uses the most expensive and most accurate algorithm.

The particle step control, kmultiple, sets a ratio between the time step for particles and the time step for fields. It can be set subject to the constraint that particles must be moved frequently enough that they do not exceed one cell in one step. If particles violate this, then they will not be tracked correctly, and the effect of the current due to their motion will not be accurately added to the electromagnetic fields. When particles are moving at speeds significantly less than the speed of light, kmultiple can be greater than unity without violating this rule. Depending on other factors such as charge density noise growth, this can save significant cpu for slow moving particles without reducing accuracy.

The relativistic kinematics option, arelopt, allows control of the use of the fully self-consistent relativistic kinematics calculation vs. the use of a simplified kinematics which ignores both magnetic fields and the relativistic factor, γ. Magnetic fields are ignored in the non-relativistic case, since it is assumed that $v \times B << E$. When applied magnetic fields are present, the relativistic option must be used. When appropriate, the non-relativistic option can save significant cpu.

Restrictions:

1. The time step ratio, kmultiple, must be chosen to preclude a particle from moving more than one cell width in a single kinematics time step.

2. Time-domain only.

See Also:

COURANT
FIELDS

LINPRINT

Function: Specifies printing two-dimensional maps of field components in scientific notation.

Format:

LINPRINT istep axaxis ayaxis iskip ivolume scales ;

Arguments:

istep	-	step interval (integer).
axaxis	-	horizontal axis of map (alpha). = X1, X2, or X3.
ayaxis	-	vertical axis of map (alpha). = X1, X2, or X3.
iskip	-	planar skip interval between maps (integer).
ivolume	-	volume indices (integers).
scales	-	scale factors for field components (nine unitless, real values). = 0, field component omitted. ≠ 0, field component plotted.

Description:

The LINPRINT command instructs SOS to print field component data as a table of numbers in scientific notation. The table displays the data more precisely, though less compactly, than a plot. The values of the field components that are within the region specified by indices, ivolume, are printed plane by plane at steps that are multiples of the volume step interval, iskip.

The field component values are multiplied by the scale factors, scales, and are printed out in scientific (exponential) notation to four significant digits. The nine scale factors correspond to scales for the nine fields, E1, E2, E3, B1, B2, B3, J1, J2, and J3, respectively. Any field for which the scale is zero is not output. The output values are arranged in a table with vertical and horizontal axes arranged according to axaxis and ayaxis. The values corresponding to a single grid index of the table vertical axis are printed on a single line or, if more than ten values, a series of lines. To the left of these values is printed that corresponding grid index.

Restrictions:

No more than ten LINPRINT commands may be specified.

Function: Specifies printing two-dimensional maps of field components in single-integer, powers-of-ten notation.

Format:

LOGPRINT istep axaxis ayaxis iskip ivolume scales ;

Arguments:

istep	-	step interval (integer).
axaxis	-	horizontal axis of map (alpha). = X1, X2, or X3.
ayaxis	-	vertical axis of map (alpha). = X1, X2, or X3.
iskip	-	planar skip interval between maps (integer).
ivolume	-	volume indices (integers).
scales	-	scale factors for field components (nine unitless, real values). = 0, field component omitted. = ≠, field component plotted.

Description:

The LOGPRINT command instructs SOS to print field component data as a two-dimensional array of alphanumeric characters, so providing a compact display of field values over a region of space. The array is printed as a rectangular block with horizontal axis grid indices above and vertical axis grid indices to the left. Each character in the array is chosen from a set of characters corresponding to different ranges of values. At a given position in the array, the character corresponds to the range which includes the value of the field component at that position in the spatial grid.

The horizontal and vertical axes are specified by axaxis and ayaxis. The array of characters is confined to the region specified by the volume indices, ivolume, and is printed one plane at a time at time steps that are multiples of istep.

The correspondence between characters and ranges of values is determined by the entire range of values of the field component inside the specified region. Ranges are spaced logarithmically from the minimum magnitude value to the maximum magnitude value; however, each range will not be made so small that it cannot contain values with base ten logarithms differing by more than the minimum resolution factor, scale. These ranges, from lowest to highest, correspond to the digits 0 through 9, respectively. Positive field values are represented by these digits, while negative field values use these same digits with minus signs overprinted. Values of exactly zero are represented by spaces.

The nine scale factors corresponding to resolution factors for the nine fields E1, E2, E3, B1, B2, B3, J1, J2, J3. To omit a field from output, set its scale factor to zero.

Restrictions:

No more than ten LOGPRINT commands may be specified.

Function: Absorbs outgoing waves as they leave the simulation.

Format:

LOOKBACK afscale(xb, xn)* phvel asnor isurface ;

Arguments:

afscale	-	geometric scaling factor (unitless or alpha).
phvel	-	phase velocity factor, v/c (unitless).
asnor	-	surface normal alignment (alpha).
isurface	-	surface indices (six integers).

Description:

The LOOKBACK command specifies an outlet boundary condition which absorbs any outgoing waves generated from within the simulation space. The location of the boundary is defined by isurface and the surface normal alignment by asnor. The phase velocity factor of the outgoing waves, phvel, is set to some fraction of the speed of light. The geometric scaling factor, afscale, is used to scale the fields at the points next to the boundary, so that linear interpolation can be done on those fields, and the fields on the boundary. This argument may be input as a function or a number. If a function is used, the functional form must first be defined with the FUNCTION command. For cartesian coordinates, afscale is unity; for non-cartesian coordinates, afscale is a function of the coordinate conformal to the boundary using the full-grid point on the boundary and the full-grid point next to the boundary. If LOOKBACK boundaries overlap, then the phase velocity of the last-entered command is used in the overlap region.

Restrictions:

1. No more than ten LOOKBACK commands may be specified.

2. If a function is specified for afscale, it must be defined prior to being referenced

3. Time-domain only. The effect on frequency-domain is unknown.

4. All specified lookback boundaries must be in the same plane.

5. The time step, δt, must satisfy $\delta t < \dfrac{\delta x}{c \cdot phvel}$, where δx is the cell length normal to the boundary and c is the speed of light.

See Also:

FUNCTION
VOLTAGE

Reference:

B. Goplen, "Boundary Conditions for MAGIC," Mission Research Corporation Report, MRC/WDC-R-019, presented at the Twenty-Third Annual Meeting APS Division of Plasma Physics, 12-16 October 1981.

Examples:

```
FUNCTION "GSCAL(R1,R2) = R2/R1" ;
LOOKBACK GSCAL 1.0 ANTI-ALIGN 41 44 1 61 1 36 ;
```

The commands above define an outlet on a grid that extends 60 full-grid points along the x2-coordinate and 35 full-grid points along the x3-coordinate. The coordinate system is a cylindrical and the outlet is anti-aligned with the x1-coordinate.

The geometric scaling factor is defined by a function, GSCAL, that takes the ratio of the full-grid point of the x1-coordinate on the boundary, R1, and the full-grid point of the x1-coordinate next to the boundary, R2. In the definition of functions for the scaling factor, it is important to remember that the first variable, R1, is always the full-grid point on the boundary of the coordinate that is conformal to the boundary. The second variable, R2, is always the full-grid point next to the boundary of the coordinate conformal to the boundary.

```
LOOKBACK 1 1 ANTI-ALIGN 41 41 1 61 1 36 ;
```

The command given above defines the same outlet on a cartesian grid. A constant has been used instead of a function for afscale.

Function: Controls error message printing.

Format:

MESSAGE aop ;

Arguments:

aopt - print option (alpha).
= NONE, do not print error messages.
= ABORT, print abort (code) error messages.
= ERROR, print abort and user error messages.
= WARNING, print abort, user, and warning error messages (default).

Description:

This command allows the user to tell the code to print or not print messages for the different levels of errors. The levels of errors are described in the TERMINATE command. This command is provided primarily to allow warning messages to be turned off. The option to turn off ABORT and ERROR level messages should be used with extreme caution.

Restrictions:

Due to the evolutionary nature of the code, not all errors are yet categorized by severity. Messages for such errors will not be controllable from this command no matter what is set for the message option.

See Also:

PAUSE
TERMINATE

NOECHO

Function: Turns off echo of processed commands to output file.

Format:

NOECHO ;

Arguments:

None

Description:

The NOECHO command suppresses output of processed commands to the output file. This command is used in conjunction with the ECHO command. It will not, under any circumstances, suppress the output of errors or error messages.

See Also:

ECHO

Examples:

```
C This input demonstrates the use of ECHO and NOECHO.
  In the process of defining ten metallic vanes, four
  coordinate locations must be shifted, and NOECHO and
  ECHO are used to suppress output from those commands
  as unnecessary.  For the commands between the NOECHO
  and the ECHO, output of the processed command will
  only occur if there is an error in the command. ;

DO I 1,10,1 ;
  NOECHO ;                                C Turn off output.    ;
  DEFINE "IX1 =  2 + (I-1)*NXPV" ;        C Output suppressed. ;
  DEFINE "IX2 =  5 + (I-1)*NXPV" ;        C Output suppressed. ;
  DEFINE "IX3 =  9 + (I-1)*NXPV" ;        C Output suppressed. ;
  DEFINE "IX4 = 12 + (I-1)*NXPV" ;        C Output suppressed. ;
  ECHO ;                                  C Turn on output.    ;
  CONDUCTOR VANE'I' ANTI-ALIGN    IX1,NGRDY      IX2,NGRDY
                                  IX2,IVANEY     IX3,IVANEY
                                  IX3,NGRDY      IX4,NGRDY   ;
ENDDO ;
```

Function: Specifies simulation variables to be plotted as a function of time or frequency.

Format:

(for field variables)
OBSERVE istep aprint FIELD ifft tstart tend
fstart fend afield iline ;

or

(for energy variables)
OBSERVE istep aprint ENERGY ifft tstart tend
fstart fend aener iener itint ;

or

(for strut variables)
OBSERVE istep aprint STRUT astruct ifft tstart tend
fstart fend acomp iline ;

or

(for circuit variables)
OBSERVE istep aprint CIRCUIT astruct ifft tstart tend
fstart fend xfact isurface ;

Arguments:

istep - step interval (integer).

aprint - printed output option (integer).
= NO.
= YES.

astruct - name of strut or structure for strut and circuit options (alpha).
= name of strut selected.
= name of structure selected for integral.

ifft - fourier transform (fft) option (integer).
= 1, data only, no fft.
= 2, fft only, real and imaginary parts.
= 3, fft only, magnitude.
= 4, data and fft, real and imaginary parts.
= 5, data and fft, magnitude.

tstart,tend - time boundaries for fft integration (sec).

fstart,fend - frequency boundaries for fft plot (hz).

afield - field component option flag (alpha).

= E1.	= XROE.
= E2.	= XRON.
= E3.	= SG*E1.
= B1.	= SG*E2.

		= B2.	= SG*E3.
		= B3.	= J1*SG*E1.
		= J1.	= J2*SG*E2.
		= J2.	= J3*SG*E3.
		= J3.	= CIRCUIT, B-FIELD integral.
		= SG.	
aener	-	energy components option (integer). = CREATED, created particle energy. = DESTROYED, destroyed particle energy. = SURVIVING, surviving particle energy. = GAINED, particle energy gained from fields. = VOLTAGE, voltage boundary energy (net loss). = LOOKBACK, lookback boundary energy (net loss). = BOUNDARY, total boundary energy (net loss). = ELECTRIC, electric field energy. = MAGNETIC, magnetic field energy. = EM, electromagnetic field energy. = TOTAL, total energy. = RESIDUAL, residual energy. = RATIO, residual to total ratio.	
acomp	-	strut component option flag (integer). = VOLTAGE. = CURRENT.	
xfact	-	sign and multiplicative factor for integral.	
iener	-	species/surface/field component index (integer). = 0, sum energies over index. > 0, choose indexed energy.	
itint	-	energy integration option (integer). = 0, rate of change in energy. = 1, energy.	
iline	-	line indices (six integers).	
isurface	-	surface indices (six integers).	

Description:

The OBSERVE command instructs SOS to measure and record specified simulation variables during the simulation and produce time history plots, i.e., plots of the variables versus time at the end of the simulation. Observable simulation variables are of four types: electromagnetic fields, energy components, strut quantities, and circuit integrals around loops, which in effect compute surface currents. The type chosen determines the format of the command. During the simulation, SOS periodically locates or calculates the desired variables

and stores them on disk. At the end of the simulation, SOS reads in the complete time history, which can be printed, Fourier transformed, and plotted.

The frequency with which SOS measures simulation variables is specified by istep, the number of the steps between measurements. A small value for istep results in time history plots with closely spaced points or Fourier plots with high frequencies. When aprint is given the argument, YES, time histories are printed as well as plotted, resulting in voluminous output for long simulations. Note that both istep and aprint apply to all time histories in the simulation, so when two or more OBSERVE commands are present, SOS uses aprint and istep arguments from the last OBSERVE command.

For [illegible] variables, the simulation variable is specified by the field component, afield, and the line indices, iline... . Six indices, rather than three, are used to specify position, allowing the option of integrating the field component between two points. (If the line has zero length, no integration is performed. If the pair of indices differ, then the specified field component will be integrated along the line segment.)

The energy component, aener, is qualified by two additional arguments. The first, iener, indexes a particular species, boundary, or component when observing particle energy, boundary flux, or field energy, respectively. If iener is set to zero, the energies corresponding to all valid values of iener are summed. For some values of aener, such as EM and RESIDUAL, iener must be zero. The second, itint, selects either a rate of change in the energy variable, or an accumulated energy. Note that some energies are instantaneous, such as electric field energy, while others are accumulated over time, such as energy lost through lookback boundaries. Rates of change are available for either type of energy variable.

The strut option allows measurement of voltage and current variables which result from use of a STRUT command. The circuit option provides the capability to perform line integrals of magnetic fields surrounding some structure. The scaling factor, xfact, allows the line integral to be converted to units of current.

The time histories may be transformed into the frequency-domain. The Fourier transform option, ifft, specifies the combination of plots desired for any simulation variable. The time boundaries, tstart and tend, specify an interval over which Fourier integration is performed; setting both to zero selects the entire time history. The frequency boundaries, fstart and fend, specify limits for the plots of the transformed data.

Restrictions:

1. The total number of time histories measured in a simulation is limited to twenty.

2. The line indices specified for integration of a field component must be conformal.

3. After the simulation is ended, time history data is loaded into the memory space used to hold field components. Consequently, the total number of time history records must be less than one-fourth this amount of storage.

See Also:

BALANCE
FIELDS
STRUT

Function: Pauses on errors of various severities.

Format:

PAUSE aopt ;

Arguments:

aopt - pause option (alpha).
= NONE, do not pause on errors (default).
= ABORT, pause on abort (code) errors.
= ERROR, pause on abort or user errors.
= WARNING, pause on abort, user, or warning errors.

Description:

This command allows the user to tell the code to pause on different levels of errors. The levels of errors are described in the TERMINATE command. The option to pause the code is only useful when debugging code. Hence this command should only be used when working with MRC personnel to examine problems with running the code.

Restrictions:

Due to the evolutionary nature of the code, not all errors are yet categorized by severity. Such errors will not trigger a pause no matter what is set for the pause option.

See Also:

MESSAGE
TERMINATE

PHASESPACE

Function: Plots particle phase space.

Format:

PHASESPACE istep axaxis ayaxis aspecies
volume pvolume ;

Arguments:

istep	-	step interval (integer).
axaxis	-	x-axis variable (alpha).
		= X1, X2, X3, P1, P2, or P3.
ayaxis	-	y-axis variable (alpha).
		= X1, X2, X3, P1, P2, or P3.
aspecies	-	species option (integer or alpha).
		= ALL.
		> 0, a particular species index.
volume	-	X1 coordinate window boundaries (m).
	-	X2 coordinate window boundaries (m or rad).
	-	X3 coordinate window boundaries (m or rad).
pvolume	-	P1 coordinate window boundaries (m/sec).
	-	P2 coordinate window boundaries (m/sec).
	-	P3 coordinate window boundaries (m/sec).

Description:

The PHASESPACE command makes plots of particle positions in phase space at times specified by the time-sequence index, itime. The coordinate or momentum along the x-axis, axaxis, and the y-axis, ayaxis, specify the nature of the phase-space plot. Multiple plots are produced, one for each species by use of the argument, ALL, for the species type, aspec. Otherwise, the species is specified by indices representing the order in which the species were introduced in PARTICLE commands. The coordinates, volume and pvolume, specify a window in phase space. Particles within the window are plotted; those outside are excluded. All of the coordinate windows should be specified regardless of which coordinates are actually being plotted.

Restrictions:

Time-domain only.

Examples:

The argument istep is set to 100 to plot phasespace every 100 time steps. The x1-coordinate is plotted on the x-axis, and the x2-coordinate is plotted on the y-axis. Plots for all particle species are produced. The window is given as:

x1-coordinate range:	0.0 to 0.128 meters.
x2-coordinate range:	0.025 to 0.05 meters.
x3-coordinate range:	0.025 to .05 meters.
p1-momentum range:	-9×10^9 to $+ 9 \times 10^9$ m/sec.
p2-momentum range:	-9×10^9 to $+ 9 \times 10^9$ m/sec.
p3-momentum range:	-9×10^9 to $+ 9 \times 10^9$ m/sec.

Note that momentum windows are specified, even though only spatial coordinates are plotted. The command is

```
PHASESPACE 100 X1 X2 ALL 0.0 0.128 0.025 0.05 0.025 0.05
     -9E9 9E9 -9E9 9E9 -9E9 9E9 ;
```

PHOTON

Function: Specifies photon sources.

Format:

PHOTON amodel aspace(x) atime(t) tadvance iwave x1so x2so x3so ;

Arguments:

amodel	-	photon source name (alpha). = arbitrary, user-defined in FUNCTION command.
aspace(x)	-	spatial function name (alpha). = arbitrary, user-defined in FUNCTION command.
atime(t)	-	temporal function name (alpha). = arbitrary, user-defined.
tadvance	-	retarded time advancement (m).
iwave	-	wave-front option (integer). = 100, plane wave in x1. = 010, plane wave in x2. = 001, plane wave in x3. = 110, cylindrical wave in x1,x2. = 011, cylindrical wave in x2,x3. = 101, cylindrical wave in x1,x3. = 111, spherical wave in x1,x2,x3.
x1so	-	source location in x1 coordinate (m).
x2so	-	source location in x2 coordinate (m or rad).
x3so	-	source location in x3 coordinate (m).

Description:

The PHOTON command allows the user to specify one or more photon source models to represent emission processes. Each model so specified must be given a unique, user-defined alphanumeric name. This is the photon source model, amodel; acceptable arguments might include such labels as SOURCE, PHOTONS, X-RAYS, or anything else that has relevance to the user. The photon source model will also be referred to by the EMISSION command, which fully specifies the emission model.

The photo-emission process in SOS is described by the equation,

$$\frac{d^5q}{dA\,dt\,dE\,\sin\theta\,d\theta} = \frac{1}{\pi}\eta s(t)f(E)\cos\theta\,,$$

where

$$\eta = -41.87(\eta'/\xi)\chi(r).$$

In the PHOTON command, the spatial function, aspace (x), and the temporal function, atime (t), represent the parameters $\chi(r)$ and s(t), respectively. Both must be supplied using FUNCTION commands. For both functions, the spatial distance is computed from the photon source origin, (x1s0, x2s0, x3s0), measured in meters, to the point of local emission. The geometrical nature of the photon output is specified by the wave-front option, awave. The regarded time advancement, tadvance, should be used initially to position the wave front near the emitting structure.

Restrictions:

1. Not relevant to frequency-domain.
2. The number of PHOTON commands in a simulation is limited to five.
3. The temporal function must be normalized to unity.

POISSON

Function: Specifies an electrostatic solution at a boundary surface.

Format:

POISSON ncond [aname volts,...] axis coef relax iters isurface ;

Arguments:

ncond	-	number of conductors (integer).
aname volts	-	pairs of structure names (alpha) and potentials (V).
axis	-	coordinate with largest gradient (alpha).
coef	-	linear coefficient in Poisson equation (m**-2).
relax	-	over-relaxation coefficient, omega (unitless).
iters	-	maximum number of iterations (integer).
isurface	-	surface indices (six integers).

Description:

The POISSON command is used to specify a solution of the generalized Poisson equation to obtain an electrostatic field distribution and will produce values for the scalar potential as well as the components of electric field. This solution may be used to construct an input TEM wave at a surface of the simulation volume for a subsequent transient simulation. The numerical solution is obtained by successive over-relaxation (SOR). The user-specified maximum number of iterations will be performed, which determines the accuracy of the solution. The resulting electric field distribution will be curl-free, within numerical limitations.

The arguments in the POISSON command begin by specifying the number of unique conductors, ncond, (unique means unique by name) followed by the name, acond, and potential, volts, for each. The largest gradient coordinate, axis, is used in making an initial estimate of potential prior to iteration. Proper choice of arguments, X1, X2, or X3, will enhance convergence.

The results of the POISSON solution are automatically stored in the appropriate components of the static fields, E1ST, E2ST, or E3ST. There they may be accessed by a matching VOLTAGE command to inject a wave with the TEM profile computed here.

To view a POISSON solution, use a VOLTAGE command to inject the solution onto the dynamic electric fields, run for one time step, and output the dynamic fields at the location of the VOLTAGE boundary.

Restrictions:

Conductors specified as unique potential surfaces must be defined in STRUCTURE commands.

See Also:

VOLTAGE
STRUCTURE
SYMMETRY

PRESCRIBE

Function: Prescribes analytical fields for particle kinematics.

Format:

PRESCRIBE apopt afield af(t) af(x1) af(x2) ;

Arguments:

apopt	-	product function option (alpha). = YES or NO.
afield	-	field component (alpha).
af(t)	-	temporal function (alpha). = arbitrary, user-defined in FUNCTION command.
af(x1)	-	x1-coordinate function (alpha). = arbitrary, user-defined in FUNCTION command.
af(x2)	-	x2-coordinate function (alpha). = arbitrary, user-defined in FUNCTION command.
af(x3)	-	x3-coordinate distribution (alpha). = arbitrary, user-defined in FUNCTION command.

Description:

The PRESCRIBE command provides a way to define the electric and magnetic fields governing particle motion analytically. Options include a triple product function (t, x_1, x_2, and x_3) or user-generated code.

Without PRESCRIBE, the kinematic forces on particles in SOS are computed from the interpolation of dynamic fields at the particle position. The PRESCRIBE command replaces the role of the dynamic fields in particle kinematics with fields obtained from analytical functions. The command has two options, a triple product of functions specified by the user in the command (aprod = YES) and complete user specification of fields from user-generated code (aprod = NO). The triple product function is given by

$$f = af(t) \cdot af(x1) \cdot af(x2) \cdot af(x3) .$$

In the other case, user-generated code must be inserted into subroutine PRESCRIBE.

Function: Presets electric and magnetic fields.

Format:

```
PRESET afield CYLINDRICAL afunction(r,theta,z)
          [SCALE scale]
          [MODIFY amodify] ;
PRESET afield CARTESIAN afunction(x,y,z)
          [SCALE scale]
          [MODIFY amodify] ;
PRESET afield READ afile aelement aformat
          [SCALE scale] ;
```

Arguments:

afield	-	field component (alpha).
amodify	-	field modification option (alpha). = ADD, adds to previous value. = REPLACE, replace previous value.
afunction	-	spatial distribution of field (alpha). = arbitrary, user-defined in FUNCTION command.
afile	-	file name (alpha).
aelement	-	source field name (alpha).
aformat	-	file data type (alpha). = ASCII or BINARY.
scale	-	scaling factor (real).

Defaults:

The default values for unspecified arguments are listed below.

Keyword	Arguments	Value
MODIFY	amodify	REPLACE
SCALE	scale	1.0

Description:

The PRESET command initializes or modifies selected electromagnetic field components to user-specified values. The fields which may be set are E1, E2, E3, B1, B2, B3, E1ST, E2ST, and E3ST. The dynamic field refers to the fields, E1-B3. The static fields, E1ST, E2ST, and E3ST, are used by the VOLTAGE command to apply a voltage to a specified surface. For this

use, the preset command must set the components that are tangential to the voltage surface (normal to a wave entering through the voltage boundary). The static fields are thus located in three dimensions by the VOLTAGE command that will be using the data.

The CARTESIAN and CYLINDRICAL options can each be used for simulations defined in both cylindrical and cartesian coordinates. The options simply change the meaning of the arguments of the function used to specify the field. The function specifies the value of the field over the three-dimensional space. Thus, the specified fields will be assigned values at each location specified by the function.

The READ option reads data in the same format written by the MAGIC and SOS codes using the DUMP and DATAFILE commands. The field data must be SURFACE type data written by DUMP, except for the DYNAMIC field option which must be a field record as written by the DATAFILE command. Since surface data is two-dimensional, the surface data is applied at each X3, for any of the three-dimensional fields E1-EB. In all cases, the grid definition for the data read must match the grid definition of the simulation. No interpolation is made.

For those occasions when it is necessary to specify a field as the superposition of fields, the argument, amodify, should be set to ADD rather than REPLACE. When the ADD option is selected, the fields are added to the existing fields established by earlier PRESET commands. Use of REPLACE will cause the new field values to replace the previous values.

Restrictions:

The read option does not interpolate, so the read record must have the same grid definition as the current simulation.

See Also:

FUNCTION
POISSON
VOLTAGE

Examples:

1. The following command sequence sets up a VOLTAGE boundary conformal to X3 to inject a TM mode.

```
C Set mode numbers for mode.   ;
DEFINE NORDER = 22 ;
DEFINE NZERO = 2 ;
C Define functional form of radial and angular
```

```
   fields for TM mode. ;
DEFINE RKC = BESSELJNZ(NORDER,NZERO) / XDIST ;
FUNCTION ERAD(R,THET) =
BESSELJP(NORDER,RKC*R)*COS(NORDER*THET) ;
FUNCTION ETHET(R,THET) =
BESSELJN(NORDER,RKC*R)*SIN(NORDER*THET) ;

C Resolve fields into X and Y components for cartesian coordinates;
          FUNCTION EX(R,THET) =
ERAD(R,THET)*COS(THET) - ETHET(R,THET)*SIN(THET) ;
          FUNCTION EY(R,THET) =
ERAD(R,THET)*SIN(THET) + ETHET(R,THET)*COS(THET) ;
```

C Set the static fields for the tangential components of the electric field to the functional form for the desired input wave. During the simulation, the static fields are multiplied by the time-dependent function specified in the VOLTAGE command and are then injected at the location specified by that command. ;

```
PRESET E1ST CYL EX ;
PRESET E2ST CYL EY ;

C Specify location at which to inject TM wave. ;

FUNCTION TVOLT(T) = 1.E16 * SIN( 1.E8 * T ) ;
VOLTAGE FIELD TVOLT 0 0 .5 ALIGN  1 NGRDX  1 NGRDY    1 1 ;
```

The following example demonstrates the injection of a cartesian TE mode onto an X2-X3 face of a simulation.

```
C Set mode numbers for mode. ;

DEFINE NX = 15 ;
DEFINE NY = 15 ;
```

C Define functional form of the tangential fields for the desired TE mode. ;

```
DEFINE PI = 3.1415926;
DEFINE RKX = NX*PI/XDIST;  DEFINE RKY = NY*PI/YDIST;
FUNCTION EXFUNC(Z,X,Y)  = COS(RKX*X)*SIN(RKY*Y);
FUNCTION EYFUNC(Z,X,Y)  = SIN(RKX*X)*COS(RKY*Y);
```

C Set the static fields for the tangential components of the electric field to the functional form for the desired input wave. ;

```
PRESET E2ST CART EXFUNC ;  PRESET E3ST CART EYFUNC ;
```

C Define boundary at which to inject TE wave.;

```
FUNCTION TVOLT(T) = 1.E16 * SIN( 1.E8 * T ) ;
VOLTAGE FIELD TVOLT 0 0 .5 ALIGN  1 1    1 NGRDX  1 NGRDY    ;
```

The following command is used to add to the B3 field the values obtained from the function BFIELD.

```
DEFINE BMAGN 0.1 ;
FUNCTION BFIELD(X,Y,Z) = BMAGN * (X*X +Y*Y + Z*Z) ;
PRESET B3 CARTESIAN BFIELD MODITY ADD ;
```

Function: Specifies simulation variables to be plotted as functions of a linear coordinate.

Format:

RANGE atimer iprocess fstart fend afield iline ;

Arguments:

atimer	-	timer name (alpha).
iprocess	-	plot control option (integer).
		= 1, plot y(x).
		= 2, plot FFT of y(x), real and imaginary parts.
		= 3, plot FFT of y(x), magnitude only.
		= 4, plot y(x) and FFT, real and imaginary parts.
		= 5, plot y(x) and FFT, magnitude only.
fstart,fend	-	frequency boundaries for fft plot (cyc/m).
afield	-	field component (alpha).
		= E1, ..., J3.
iline	-	line indices (six integers).

Description:

The RANGE command instructs SOS to plot a field component as a function of position along any conformal line in the simulation. The data may optionally be transformed to a function of spatial frequency.

Range plots are made at times specified through the timer, atimer. The data plotted is Fourier transformed if iprocess is greater than unity. The transformed data is plotted as a function of spatial frequency which has units of cycles per meter. The frequency boundaries, fstart and fend, specify limits for the plots of the transformed data; setting both to zero displays frequencies from zero to one-half the smallest grid cell length.

Plots are made of the value of the field component, afield, along the line (or range) specified by iline, which must be conformal.

Restrictions:

The total number of RANGE commands defined in a simulation is limited to ten.

RECORD

Function: Records the simulation to disk for later continuation.

Format:

RECORD istep afile iretain ;

Arguments:

istep	-	record step interval (integer).
afile	-	output file (alpha). = arbitrary, user-defined.
iretain	-	retention period (days).

Description:

The RECORD command provides protection against the loss of the simulation results in case the computer goes down during the run. The time-sequence index, istep, is used to specify the time steps at which the simulation is recorded. The RECORD command also enables saving a simulation at the end of a run in case continuation to longer times may be later desired. To do this, simply set the time step indices to record the simulation after the last time step of the current run.

The RECORD command writes only one file per simulation, no matter how many times it writes out the simulation. Each recording overwrites the previous recording. The command does start a new version of the file for each execution of SOS, so that files that exist at the beginning of an execution of SOS will not be overwritten during execution.

The retention period for the file must be input on all machines, but is ignored except on the CRAY. On the CRAY, files will be deleted automatically by the operating system at the end of the retention period.

To continue a simulation from a file written by the RECORD command, use the RESTART command.

See Also:

RESTART

Function: Reads in a previously recorded simulation for continuation.

Format:

RESTART afile ;

Arguments:

afile - input file (alpha).
= arbitrary, user-defined.

Description:

The RESTART command reads the specified file immediately at the time that the command is processed. Thus, all information previously entered into the simulation via commands preceding the RESTART command is overwritten. The simulation will be continued from where it left off, and even the output will continue as specified in the original simulation.

If the original simulation ran to completion and continuation is desired, a FIELDS command must be entered after the RESTART command; it must include a maximum number of time steps greater than that recorded in the file read by the RESTART command. If the original simulation was terminated prematurely by a computer crash or a TIMEOUT command, then no FIELDS command is required.

The RESTART command does not initiate the simulation. After the RESTART command, other commands (such as the FIELDS command) may be entered. The simulation is started, as for any simulation, with the START command.

Restrictions:

1. Certain commands should not be entered after a RESTART command and may either be ignored or may damage the fidelity of the simulation. Commands that change the physical configuration being simulated should not be entered. Similarly, changing the field algorithm may damage the simulation. Most commands that request output may be entered if additions to the output are desired. The deletion of output is not possible.

2. The OBSERVE command should be used carefully. It is not possible to retroactively request information about time steps preceding the time step at which the continuation is being restarted. Thus, OBSERVE commands may be added, but they should only request observation of time steps after the continuation of the simulation.

See Also:

OBSERVE
RECORD

Function: Returns from a call to a command file.

Format:

RETURN [astat] ;

Arguments:

astat - option to control command file on return.
= SAVE, keep command file open.
= CLOSE, close command file.
= REWIND, SAVE plus rewind the file.

Defaults:

Argument	Default Value
astat	CLOSE

Description:

The RETURN command is used to terminate a command file entered via a CALL command. Generally, the file opened by the call will be closed on exit. However, the file can be kept open to resume processing on a subsequent call. Only one file can be kept open for future reading of commands from that file.

See Also:

CALL

Function: Snaps coordinate to the spatial grid.

Format:

SNAPGRID aaxis avar rval [aopt] ;

Arguments:

aaxis	-	coordinate axis (alpha). = X1, X2, or X3.
avar	-	name of real variable in which snapped coordinate is stored (alpha).
rval	-	coordinate value to be snapped (m or rad).
aopt	-	index option (alpha). = CELL, returns initial coordinate of cell that rval is in. = FULL, returns coordinate of nearest full-grid point. = HALF, returns coordinate of nearest half-grid point.

Description:

The SNAPGRID command returns a coordinate value based on rval and aopt. It is similar to ENGRID, except that instead of returning an index, it returns the coordinate corresponding to that index. Rval is the coordinate to be snapped. If aopt is not specified, CELL is assumed. For the CELL option, SNAPGRID returns the coordinate of the beginning of the cell that rval is in. For FULL, the coordinate of the nearest full-grid point is returned, and for HALF, the nearest half-grid coordinate is returned. The snapped coordinate is returned in avar, which must be a real variable. If avar is an integer variable, a warning will be printed and the result will be truncated.

Restrictions:

The axis must be defined with an XnGRID command before the SNAPGRID command is used. Otherwise, an error will occur. If rval lies outside of the defined grid, the lowest or highest index is the one that is converted, depending on whether rval is below or above the defined grid, respectively.

See Also:

ENGRID
UNGRID
XnGRID

Examples:

1. The following command finds out what cell on the X1-axis that 3.71E-3 is located in and the real value for the lower edge of that cell is placed in RCELL.

```
SNAPGRID X1 RCELL 3.71E-3 CELL ;
```

2. The following command is the same as the above except that CELL is not specified. The result is the same because CELL is the default.

```
SNAPGRID X1 RCELL 3.71E-3 ;
```

3. The following command finds the full-grid point on the X2-axis that 0.93 is nearest to and places the value of that point in IWINDOW. Note that IWINDOW will be a truncated representation of the value found because it is an integer variable.

```
SNAPGRID X2 IWINDOW 0.93 FULL ;
```

Function: Interrupts the processing of input and initiates the simulation.

Format:

START ;

Arguments:

None.

Description:

The START command tells the code to try to simulate the problem that has been defined by the previously input commands. If there have been any errors in any of the commands input prior to the START command, the simulation will be terminated.

If there have been no errors up to the processing of the START command, then the code makes some further checks on the input data and if no errors are encountered during this second phase, then the simulation is run. If there are errors, then the code terminates the simulation and continues reading data from the input file.

Self initialization is completed before reading new input, so that commands following a START command are completely independent of those preceding it. Thus, one execution can perform multiple independent simulations, one after the other.

Example:

```
TITLE "FIRST SIMULATION" ;
      .
      .
      .
START ;
TITLE "SECOND SIMULATION" ;
      .
      .
      .
START ;
STOP ;
```

Function: Specifies times at which particle statistics are collected and printed.

Format:

STATISTICS ktime ;

Arguments:

ktime - time-interval for statistics output (integer).

Description:

The STATISTICS command causes particle statistics to be collected during the simulation. These statistics include the number created, the number destroyed, the number existing, etc. This information is printed at selected intervals during the simulation; totals and averages are printed at the end of the simulation.

Particle statistics are printed at simulation times that are multiples of ktime. The particle statistics printed are:

number existing at the last measurement,
number created since the last measurement,
number destroyed since the last measurement,
number existing at present, and
error in the number of particles.

The error reveals any variation between the actual count of existing particles and the difference between creation and destruction counts. If this number is anything but zero, then the simulation is invalid.

At the end of the simulation, several additional statistics are printed out.

These are:

total number created during the simulation,
total number destroyed during the simulation,
highest number existing during the simulation, and
average number existing during the simulation.

Restrictions:

Time-domain only.

STOP

Function: Terminates execution unconditionally.

Format:

STOP ;

Arguments:

None.

Description:

The STOP command terminates execution immediately after processing the command from the input file. Any commands that follow the STOP command will be ignored.

Example:

```
TITLE "FIRST SIMULATION" ;
      .
      .
      .
START ;
TITLE "SECOND SIMULATION" ;
      .
      .
      .
START ;
STOP ;
```

Function: Specifies conformal and non-conformal perfect conductors.

Format:

```
STRUCTURE SOLIDFILL aname ;
STRUCTURE FUNCTION aname afill asys afunc(x1,x2,x3) ;
STRUCTURE CONFORMAL aname afill ivolume ;
STRUCTURE SPHERE aname afill asys point radius ;
STRUCTURE CYLINDER aname afill asys point
        apol rpolar aazi razi radius height ;
STRUCTURE BOX aname afill asys point
        apol rpolar aazi razi rotat depth width height ;
STRUCTURE PARALLELEPIPED aname afill asys [point] ;
STRUCTURE ELLIPTICAL aname afill asys center1
        center2 primdir1 prim1 sec1 primdir2 prim2
        sec2 thetai1 thetaf1 thetai2 thetaf2 ;
```

Arguments:

aname	-	structural element name (alpha). = arbitrary, user-defined.
afill	-	inversion option (alpha). = SOLID, space within figure is made solid. = VOID, space within figure is made void.
asys	-	coordinate system for generation (alpha).
afunc(x1,x2,x3)	-	structure function (alpha). = user-defined in FUNCTION command.
ivolume	-	volume indices (six integers).
point	-	coordinates (three reals in m).
apol	-	axis for polar angle (alpha). = X1, X2, or X3.
rpolar	-	polar angle of rotation (degrees).
aazi	-	axis for azimuthal angle (alpha). = X1, X2, or X3.
razi	-	azimuthal angle of rotation (degrees).
rotat	-	rotation angle about box's axis (degrees).
radius	-	sphere or cylinder radius (m).
height	-	cylinder or box height (m).
depth	-	box depth (m).
width	-	box width (m).
point1	-	coordinates of center of 1st ellipse (three reals in m).
point2	-	coordinates of center of 2nd ellipse (three reals in m).

primdir1	-	direction of primary axis of 1st ellipse (three reals or integers).
prim1	-	length of primary axis of 1st ellipse (in m) .
sec1	-	length of secondary axis of 1st ellipse (in m).
primdir2	-	direction of primary axis of 2nd ellipse (three reals or integers).
prim2	-	length of primary axis of 2nd ellipse (m).
sec2	-	length of secondary axis of 2nd ellipse (m).
thetai1	-	initial angle section of 1st ellipse (-360 to 360 degrees).
thetaf1	-	final angle section of 1st ellipse (-360 to 360 degrees).
thetai2	-	initial angle section of 2nd ellipse (-360 to 360 degrees).
thetaf2	-	final angle section of 2nd ellipse (-360 to 360 degrees).

Description:

The STRUCTURE command is used to set up the conducting boundaries of the simulation. Each different structure requires a set of coordinates that uniquely specify the structure. The afill option is used either to fill the space with a solid conductor (SOLID) or to make a hole in a conductor that is already present (VOID). Each grid cell becomes solid or void, based on the last structure command relevant to the cell. The asys parameter is the coordinate system used to specify the structure. It does not have to be the same as the system used in the simulation.

The SOLIDFILL option fills the entire grid with conducting material. Then other structure commands (with afill = VOID) can be used to put cavities in it. This is useful if most of the geometry is conducting material.

The FUNCTION option provides arbitrary user-defined shapes. The shape is the region where the specified function has a value less than zero. The specified function is evaluated at the center of each cell, and the interior region of the structure is the region where afunc<0. That region is made solid or void based on afill. The variables x1,x2,x3 represent x,y,z or r,θ,z, or r,θ,φ, depending on the selected system. The coordinate system for x1,x2,x3 is independent of the coordinate system of the grid. Overall, the FUNCTION option of this command incorporates all the other options and more.

The CONFORMAL structure sets up a conductor that is aligned with the grid. This is a square box or wall in cartesian coordinates and a solid or hollow cylinder in cylindrical coordinates. The user specifies the initial and final grid points in each direction.

The SPHERE option generates a sphere of dimension, radius, centered at the point X1,X2,X3. A spherical shell can be generated by two concentric spheres of different radii with the inner sphere generated using the VOID option.

The CYLINDER option creates a cylinder with a specified radius and height. The cylinder can be rotated in an arbitrary direction. The command requires the specification of the center of the bottom, point, the radius, and the height. In order to rotate the cylinder, two angles must be given. The first is the polar angle. The user must specify the axis from which the angle is measured (apol) and the angle (rpolar). The second angle represents an azimuthal rotation. Again the axis from which the angle is measured (aazi) and the angle in degrees (razi) must be given.

The BOX option generates a box of user-specified dimensions that can be rotated in any direction. The box requires the three coordinates of the center of the box, the three dimensions of the sides, and the three angles of rotation. Before the box is rotated, the height is measured along the z-axis, the width is measured parallel to the x-axis, and the depth is measured on the y-axis. The first angle (rpolar) is the angle between the specified axis (apol) and the vertical axis of the box. The second angle is the azimuthal angle rotation from the axis, aazi. The third angle corresponds to a rotation about the axis of the box.

The PARALLELEPIPED is generated by specifying the coordinates of one corner and the coordinates of the three adjacent corners.

The ELLIPTICAL cylinder is the most complicated of the STRUCTURE commands. An elliptical cylinder is the shape generated by connecting two ellipses. If the two primary axes of the ellipses (primdir1 and primdir2) are in different directions, the cylinder will be twisted continuously from one primary axis to the other. A cylinder can be constructed by defining the primary and secondary axis lengths equal. A cone is generated if one ellipse is a circle (prim1 = sec1) and the other is a point (prim1 = sec1 = 0). To define each ellipse, the coordinates of the center must be specified. The parameter, primdir, is the direction of the primary axes. This can be any three-dimensional vector. The code normalizes it into a direction vector. The angle between the primary axes of the two ellipses must be less than 180 degrees. To make a full 360-degree twist, the best way to do it is to add together four 90-degree twists. Once the direction vectors are given, SOS projects each ellipse onto the plane perpendicular to the cylinder axis (the line connecting [point] and [point]). The parameters prim1, sec1, prim2, and sec2 are the primary and secondary axis lengths of the first and second ellipses respectively. The last option is to generate a wedge. To specify a wedge, the initial and final angles of each ellipse

are required. Thetai1 and thetai2 are the initial angles for ellipse1 and ellipse2. Thetaf1 and thetaf2 are the final angles. The angles are measured in degrees and are measured from the primary axis. If no wedge is desired, set the initial angle to 0 and the final angle to 360.

Overall structure may be defined beyond the grid, but only the portion within the grid is simulated. Also, there is a virtual layer of cells beyond the defined grid, and, by defining that layer as a conductor, it is possible to simulate a conductor at the limit of the simulation without using cells.

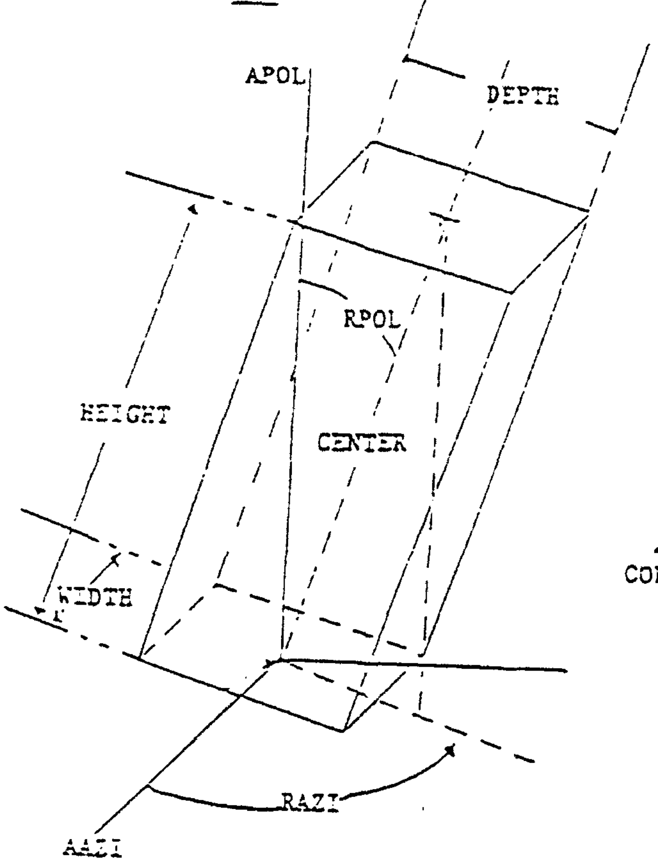
Restrictions:

The angle between the primary axes of the two ellipses in the ELLIPTICAL option must be less than 180 degrees.

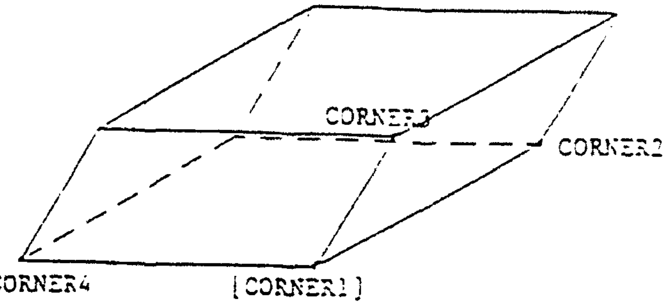
Examples:

The following figures illustrate geometries which can be generated using the STRUCTURE command.

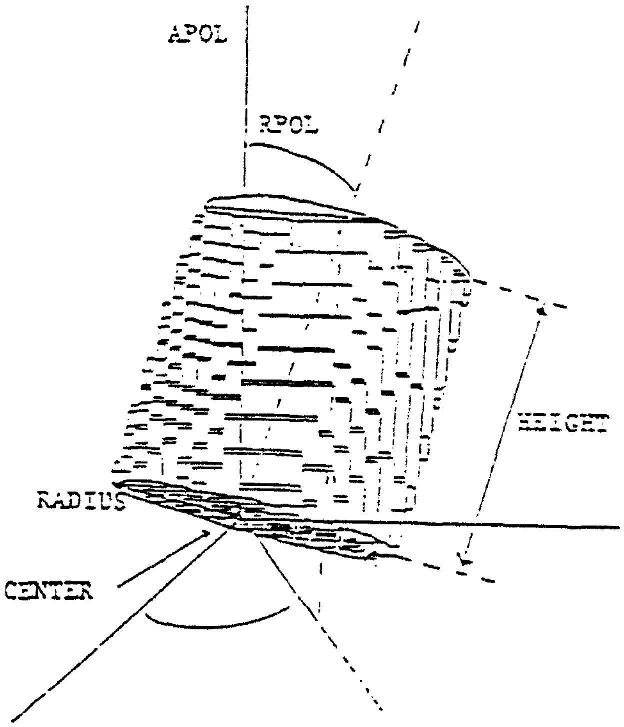
BOX



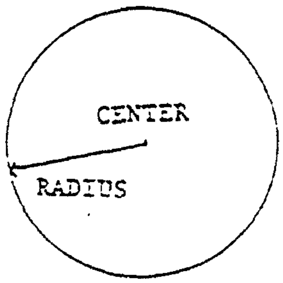
PARALLELEPIPED



CYLINDER



SPHERE



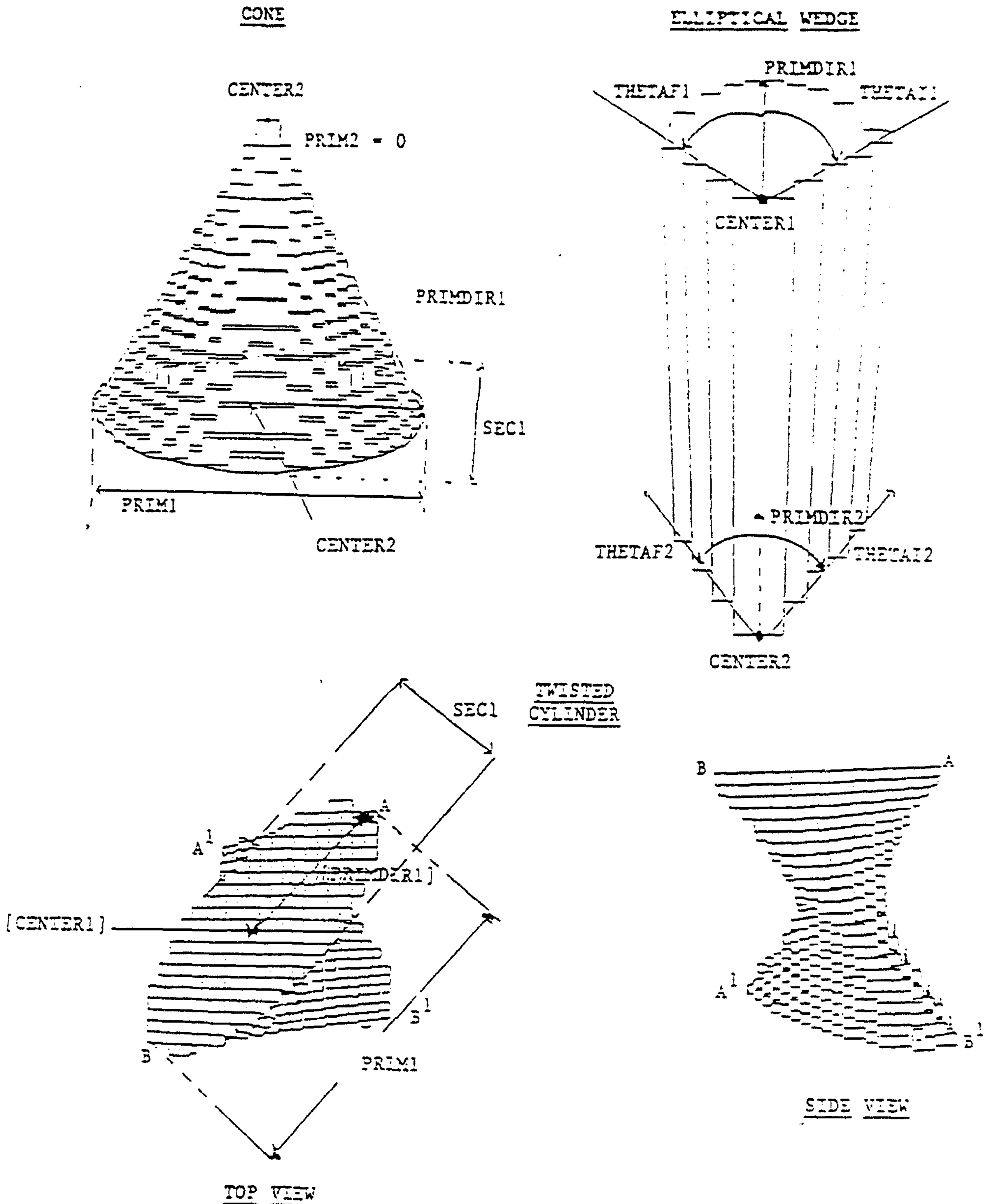


Figure 5.3. Generated geometries in SOS.

Function: Specifies strut sub-grid model.

Format:

STRUT aname radius resist kmultiple avsi(t) avsf(t) axis iline ;

Arguments:

aname	- strut name (alpha). = arbitrary, user-defined.
radius	- strut radius (m).
resist	- strut resistance (ohms/m).
kmultiple	- field-to-strut time step ratio (integer).
avsi(t)	- initial voltage function (alpha). = NULL, initial voltage is zero. = arbitrary, user-defined in FUNCTION command.
avsf(t)	- final voltage function (alpha). = NULL, final voltage is zero. = arbitrary, user-defined in FUNCTION command.
axis	- strut axis (alpha). = X1, X2, or X3.
iline	- line indices (six integers).

Description:

The STRUT command specifies the placement and properties of a thin, resistive rod with cross section too small to be modeled using the available spatial resolution. In this quasi-static approximation, the strut is modeled as a zero-by-zero-by-n-cell conductor, but with a finite radius and resistance per unit length. This treatment makes possible a strut smaller than a cell width and with properties independent of spatial grid size.

Each strut requires a unique label, aname. The strut radius is specified in meters, and the resistance is specified in ohms per meter. The time step ratio, kmultiple, specifies the number of strut time steps for each electromagnetic fields time step. Axis is the spatial grid axis to which the strut is parallel. The indices, iline, specify the position of the strut and must specify a line parallel to the specified axis.

If either end of the strut is not grounded, it is possible to apply a voltage pulse directly. Time-varying voltages, avsi(t) and avsf(t), are applied to the initial and final coordinates of the strut, respectively. If the ends of the strut are grounded, i.e., they touch conductors, the voltage functions may be set to NULL.

Restrictions:

1. The number of struts is limited to ten.
2. The sum of the lengths (in cells) of all struts is limited to 100 cells.
3. Struts must be conformal; they cannot cut diagonally across cells.
4. Time-domain only.

See Also:

FIELDS
FUNCTION

Function: Controls output of processed commands in the output file.

Format:

SUPPRESS asopt ;

Arguments:

asopt - suppression option (alpha).
= YES or NO.

Description:

The SUPPRESS command suppresses output of the processed commands to the output file. It will not, under any circumstances, suppress the output of errors or error messages, nor will it suppress the effect of error on the termination of the simulation. The command simply suppresses output from commands which, as far as SOS can determine, are correct.

The SUPPRESS command acts immediately and can be used multiple times in a single input file alternately to suppress and enable the output of correct commands in certain parts of the input file.

SYMMETRY

Function: Specifies symmetry boundaries.

Format:

SYMMETRY asym asnor isurface [asnor isurface] ;

Arguments:

asym	- symmetry option (alpha). = AXIAL, MIRROR, or PERIODIC.
asnor	- surface normal alignment (alpha).
isurface	- surface indices (six integers).

Description:

The SYMMETRY command imposes the conditions necessary to enforce certain types of symmetry on specified boundaries of the simulation space. The symmetry boundaries can be applied in any number of combinations when appropriate. They affect particle transformations as well as the electromagnetic fields. The choices available are axial, mirror, and periodic boundaries.

The axial boundary must be used if, and only if, the coordinate system is cylindrical. If the volume ($0 \le r \le R_f$) is used in place of the surface ($R_f = 0$), then a filter will automatically be applied to fields within radius R_f. The allowable time step will be substantially increased, consistent with the larger cell dimensions at R_f.

The mirror boundary, as the name suggests, replicates bilateral symmetry, which means simply that whatever occurs in the simulation space, also occurs simultaneously in the virtual space opposite the boundary. The periodic boundary is used to effect a repetitive structure or to close a system in cylindrical coordinates for which the full range of azimuth (2π) is required. Note that the periodic boundary requires two distinct surfaces which have opposite values for the surface normal alignment.

Restrictions:

1. The symmetry boundaries must be compatible with the models used in the simulation to retain physical fidelity. For example, the mirror and periodic boundaries should only be used on flat surfaces (not curved), the periodic boundaries must appear on opposite sides, and so on.

2. Periodic boundaries must not be conformal to z, i.e., periodicity in z cannot be modeled with these boundaries.

See Also:

COURANT

SYSTEM

Function: Specifies the coordinate system.

Format:

SYSTEM asyst ;

Arguments:

asyst - coordinate system (alpha).
= CARTESIAN, Cartesian (x,y,z) [Default].
= CYLINDRICAL (r,θ,z).
= SPHERICAL (θ,ϕ,r).

Description:

The three coordinate systems described above are available. If this command is not invoked, the cartesian system is assumed.

Restrictions:

1. The axial symmetry boundary condition is not automatically provided for the cylindrical coordinate system. It must be specified explicitly with an axial SYMMETRY command.

2. Periodicity in cylindrical coordinates between $\theta = 0$ and $\theta = 2\pi$ and in spherical coordinates between $\phi = 0$ and $\phi = 2\pi$ is not automatic, and should be specified with a periodic SYMMETRY command.

3. Axial boundary conditions at $\theta = 0$ and $\theta = \pi$ in spherical coordinates have not been implemented.

See Also:

SYMMETRY
X1GRID
X2GRID
X3GRID

Function: Specifies which particles are plotted on output graphics.

Format:

TAGGING ftag ;

Arguments:

ftag - fraction of particles tagged for output (unitless). $0 < \text{ftag} < 1$.

Description:

The TAGGING command specifies the fraction of particles chosen to appear in particle trajectory and phase-space plots. By default, ftag is 1.0, and particle plots will show all particles in the simulation. However, this default may produce an extremely dense plot and an extremely large plot file; setting ftag to a fraction less than 1.0 limits trajectory plots to showing only this fraction of the particles (randomly chosen) in the simulation.

TERMINATE

Function: To terminate on errors of various severities.

Format:

TERMINATE aopt ;

Arguments:

aopt - termination option (alpha).
= NONE, do not terminate on errors (default).
= ABORT, terminate on abort (code) errors.
= ERROR, terminate on abort or user errors.
= WARNING, terminate on abort, user, or warning errors.

Description:

This command allows the user to control termination of the code on different levels of errors. There are three levels of errors listed in order of decreasing severity:

ABORT, ERROR, WARNING.

ABORT conditions are generally due to code errors. ERROR conditions are caused by user-input errors that result in failure of the code. WARNING conditions are caused by user input that is legal, but which generally cause some violation of the intended usage of the code. Setting termination for errors of a particular severity, causes termination for all errors of the specified and greater severity.

Restrictions:

Due to the evolutionary nature of the code, NOT all errors are yet categorized by severity. Such errors will not trigger termination no matter what is set for the termination option.

See Also:

MESSAGE
PAUSE

Examples:

The FUNCTION and DEFINE commands may have errors which do not unconditionally abort execution. If variables are not defined prior to being used, they will generate error messages, but may not trigger an abort condition. Setting the TERMINATE command to abort on encountering an ERROR will prevent a job from continuing until the error-causing conditions are corrected. In the following example, the variable VOLTMAX has been misspelled. Thus,

this variable is undefined in the function VOLTIME. In such cases, it is better to ensure that an abort occurs prior to the simulation beginning.

```
TERMINATE ERROR ;
DEFINE VOTLMAX 50E3 ;
DEFINE WOMEGA 10.7E10 ;
FUNCTION "VOLTIME(T) = VOLTMAX * SIN ( WOMEGA*t )" ;
```

TIMEOUT

Function: Sets aside CPU time for final output.

Format:

TIMEOUT seconds areopt afile ;

Arguments:

seconds	-	CPU seconds for output (sec).
areopt	-	record option (alpha). = YES or NO.
afile	-	restart file (alpha). = arbitrary, user-defined.

Description:

The TIMEOUT command provides the opportunity to obtain output even if the amount of CPU requested for the simulation is insufficient for completion. When the remaining CPU time is equal to the argument, seconds, the code will complete the current time step and start output. Therefore, to guarantee that the output is completed, the user must allot enough time to complete both one time step and the output. A typical, conservative value is 30 seconds.

As an option, the TIMEOUT command also allows creation of the restart file, independent of requests made by the RECORD command. If the record option is selected, the code will output a restart file with the specified file name. If the record option is not selected, no file name should specified.

Restrictions:

If a filename is specified in commands TIMEOUT and RECORD, then the filename last entered will be used.

See Also:

RECORD
RESTART

Examples:

A SOS simulation may be continued in a separate execution by means of the restart feature. A restart file can be created with the TIMEOUT command. This command allows SOS to terminate gracefully and write a restart file when a CPU time limit threatens to end the simulation prematurely.

```
TIMEOUT 30 YES FAILSAFE ;
```

This command creates a RESTART file called FAILSAFE if the code determines that less than 30 CPU seconds remain for execution. Execution may be resumed at the point of termination and the simulation completed by using the following commands:

```
RESTART FAILSAFE ;
START ;
STOP ;
```

TIMER

Function: Defines a time sequence for events such as output.

Format:

TIMER atimer DISCRETE ktime [,...] ;
TIMER atimer PERIODIC kstart kstop [kstep] ;

Arguments:

atimer	-	timer name (alpha). = arbitrary, user-defined.
ktime	-	discrete time indices (integer).
kstart	-	starting time index (integer).
kstop	-	stopping time index (integer).
kstep	-	time index step (integer).

Description:

The TIMER command allows the definition of named time sequences that can be used elsewhere to trigger various events to occur on the time steps specified. Once a timing sequence is defined here, it is used by using its name in any command where an argument, atimer, is expected.

For the discrete option, a complete list of the time steps in the sequence is input. This is useful when the time steps are scattered aperiodically in time.

When the periodic option is used, three integers are input to specify the periodicity: a beginning time step, an ending time step, and a time step interval. This option functions like a Fortran do-loop, except that the time step must be positive.

The input of a time step is optional. The default value for kstep is one.

TIMER upper limit cannot be reset after a RESTART; hence, it is wise to use a generously large upper limit when the time trigger must continue after a RECORD-RESTART.

Examples:

The following example requests output of E1, B2, and B3 every 20 time steps, starting on time step 10 and continuing through time step 90. Printout will occur on time steps 10, 30, 50, 70, and 90. Printout does not occur on time step 100.

```
TIMER PTIMER  PERIODIC  10 100 20 ;
LINPRINT PTIMER E1 ... ;
LINPRINT PTIMER B2 ... ;
LINPRINT PTIMER E3 ... ;
```

TITLE

Function: Specifies a unique identification for the simulation.

Format:

TITLE "atitle" ;

Arguments:

atitle - simulation title (alpha).
= arbitrary, user-defined.

Description:

The TITLE command specifies an alphanumeric label which can be used to identify uniquely a particular simulation. This title, consisting of the alphanumeric string between the two double quotes, will be reproduced on all code output. In the event the command is not used, a blank title will appear by default on all output.

Restrictions:

1. The characters, &, #, >, and <, should not be used with DISSPLA graphics.

2. All special characters may be used with NCAR graphics; however, only upper-case letters may be used.

Function: Specifies graphical output of particle trajectories.

Format:

TRAJECTORY ktime kduration axaxis ayaxis CONFORMAL ivolume ;
TRAJECTORY ktime kduration axaxis ayaxis IMAGE volume ;

Arguments:

ktime	- time step interval (integer).
kduration	- plot duration in time steps (integer).
axaxis	- first coordinate (alpha). = X1, X2, or X3.
ayaxis	- second coordinate (alpha). = X1, X2, or X3.
ivolume	- volume indices (six integers).
volume	- volume (six reals in m).

Description:

The TRAJECTORY command will plot particle trajectory paths over a time period specified by the plot duration, kduration. The x and y axes of the plots are selectable via axaxis and ayaxis. The plotted area is designated either in grid cell or in cartesian coordinates. All particles in the specified volume are plotted.

UNGRID

Function: Converts grid index into real coordinate.

Format:

UNGRID aaxis avar ival [aopt] ;

Arguments:

aaxis	-	coordinate axis (alpha). = X1, X2, or X3.
avar	-	name of real variable to store result in (alpha).
ival	-	grid index to be converted (integer).
aopt	-	index option (alpha). = FULL, returns coordinate for full-grid value. = HALF, returns coordinate for half-grid value.

Description:

The UNGRID command returns a real value based on ival and aopt. If aopt is not specified, FULL is assumed. The real variable, avar, must be a legitimate real variable. For the FULL option, UNGRID returns the value of the full-grid point associated with ival, and for the HALF option, UNGRID returns the value of the half-grid point.

Restrictions:

The axis must be defined with an XnGRID command before the UNGRID command is used. If ival lies outside of the defined grid, the lowest or highest defined index is given, depending on whether ival is below or above the defined grid, respectively.

See Also:

ENGRID
SNAPGRID
XnGRID

Function: Specifies vector plots of two-dimensional fields.

Format:

VECTOR atimer afield
  axaxis ayaxis surface nvcx nvcy [asystem] [ascale] ;

Arguments:

atimer	-	timer name (alpha).
afield	-	field (alpha). = E, electric field. = B, magnetic field. = J, current density.
axaxis	-	horizontal axis of plot (alpha). = X1, X2, or X3.
ayaxis	-	vertical axis of plot (alpha). = X1, X2, or X3.
surface	-	surface (six reals in m).
nvcx	-	number of vectors along horizontal (integer).
nvcy	-	number of vectors along vertical (integer).
asystem	-	coordinate system for plot axes (alpha). = CARTESIAN, translate vectors to cartesian axes. = CONFORMAL, plot vectors on spatial grid axes.
ascale	-	vector magnitude scaling (alpha). = LOG, logarithmic scaling.

Description:

The VECTOR command instructs SOS to plot vectors (with arrowheads) showing the magnitude and direction of E, B, or J, fields as a function of position in space.

Vector plots are made at time steps specified through the timer, atimer. The field components to be plotted are specified by the single argument, afield. Two arguments, axaxis and ayaxis, specify the horizontal and vertical components of the plots. The region displayed is specified by surface. This region is divided into cells of equal size, nvcx cells horizontally and nvcy cells vertically. In the center of each of these cells, a vector is plotted whose length and orientation depend on field values interpolated from field data. Ordinarily, the length of the vector is proportional to the magnitude of the field; however, if the optional parameter, ascale, is set to LOG, the length will be proportional to the logarithm of the magnitude.

In cylindrical coordinates, the conformal surface, surface, might be a cylinder (x2 vs x3), a rectangle (x1 vs x3), or a sector (x1 vs x2). Usually, the surface is plotted as a rectangle, but

in the last case, field vectors are converted to cartesian and plotted on cartesian axes. Setting asystem to CONFORMAL will suppress this conversion.

Restrictions:

1. The total number of VECTOR commands in a simulation is limited to ten.

2. Vector plots can only be produced using system graphics. If the line-printer graphics option is used, no vector plots will be produced.

See Also:

GRAPHICS

Function: Specifies three-dimensional perspective of system structure.

Format:

VIEW ashade eyepoint viewpoint vang adir ivolume ;

Arguments:

ashade	-	shading option (alpha). = PLAIN. = CELLS.
eyepoint	-	eyepoint coordinates of (three reals in m).
viewpoint	-	viewpoint coordinates of (three reals in m).
vang	-	angle of field of view (degrees).
adir	-	zenith alignment and axis (alpha). = ABOVEX1. = X2ALIGN. = X3ALIGN. = BELOWX1. = X2ANTI-ALIGN. = X3ANTI-ALIGN.
ivolume	-	portion of simulation to view (six integers).

Description:

The VIEW command generates a three-dimensional perspective plot of the structure. It can take a picture of any point in the simulation from any other point in space. The picture can be magnified or reduced. The PLAIN shading option draws the structure boundaries, while the CELLS option also draws the cell divisions. The eyepoint is the point in space from which the picture is taken. It is specified by its coordinates in meters. The eyepoint can be inside or outside the simulation, but it must be outside the volume specified by ivolume. The viewpoint is the point at the center of the view. If you were taking a picture, this would be the point on which the camera would be focused. The parameter, vang, is the angle of the field of view. This is the angle subtended by the vertical axis of the final view. This option functions as a lens. A small angle will magnify a small volume (like a telephoto lens), and a large angle will reduce a large volume (like a wide-angle lens). The volume to be looked at is specified by ivolume. This is a set of six integers specifying the initial and final gridpoints of the volume. This options allows the user to look inside the walls of the simulation at a specific portion of the structure.

Restrictions:

The eyepoint must be outside the volume specified by ivolume.

Function: Specifies a time-dependent voltage to be applied at a simulation boundary.

Format:

VOLTAGE atype avolt(t) delay afscale(xb,xn)* phvel asnor isurface ;

Arguments:

atype	-	application type (alpha). = FIELD. = no other option at this time.
avolt(t)	-	temporal distribution label (alpha). = arbitrary, user-defined in FUNCTION command.
delay	-	time delay (sec).
afscale	-	geometric scaling factor (unitless or alpha).
phrel	-	phase velocity factor, v/c (unitless).
asnor	-	surface normal alignment (alpha). = ALIGN, aligned with axis. = ANTI-ALIGN, anti-aligned with axis.
isurface	-	surface indices (six integers.)

Description:

The VOLTAGE command allows the definition of a port through which waves may enter and exit the simulation. The command provides the ability to define both the port location and the entering wave. Also, it requires the specification of some properties of the expected exiting wave. The input argument, atype, must be set to FIELD.

The incident wave is specified in terms of its spatial and temporal components separately. The multiplicative product of the two terms is the total entering wave. The function, avolt(t), along with the time delay, delay, specifies the temporal dependence of the entering wave. The spatial dependence of the incident wave is obtained from two components of the static fields, E1ST, E2ST, and E3ST, which provide the tangential fields for the particular port. The required tangential field components depend on the port direction.

The static fields may be set using PRESET and/or POISSON commands. POISSON may be used for TEM modes, when the port boundary includes multiple disconnected conductors which may be set to different voltages. The POISSON solution is automatically placed into the appropriate static fields for use by the VOLTAGE command. The PRESET command can be used to set the tangential fields functionally to known analytic solutions or using results solved for in other simulations. When the PRESET command is used, care must be taken to explicitly set both appropriate components of the static fields.

VOLTAGE

Outgoing waves may be treated by specifying the phase relative velocity, phrel, of the wave, and a geometric scaling, afscale, for the port. The outgoing wave will be reflected off the port and back into the simulation, in proportion to the error in phrel, as in a physically existent impedance mismatch. Outgoing waves may be intentionally reflected entirely off the port and back into the simulation by setting phrel to zero. The geometric scaling factor denotes the changing size of the port in the direction normal to the port. In cartesian coordinates, this factor is always unity. In other coordinate systems, it is the ratio of the size of the port to its size projected one grid cell in the direction, asnor. In cylindrical coordinates, it is unity for ports normal to z or theta. In general, it represents the compression or expansion of the port.

The location of the port is set with asnor and isurface. The port must be conformal with one of the axes and is intended to be placed at one of the boundaries of the simulation region. Placement within the main region of the simulation is not prevented, but may produce unexpected and unphysical results. The direction, asnor, indicates whether to inject the entering wave towards the positive (align) side of the port or the negative side (anti-align) side of the port.

When multiple VOLTAGE commands are used, the ports must all be conformal to the same axis. It is always possible to place multiple VOLTAGE commands on the same conformal plane of the simulation. Further, ports in the same plane may overlap if their spatial dependencies are the same. In such cases, the wave specified by the last-entered VOLTAGE command will predominate in the overlap region. This latter feature is useful for waves entering dielectrics, especially for allowing outgoing waves to escape since the phase velocity is different inside and outside the dielectric, and even on its boundary. Ports may be in separate, parallel planes if their projections onto a single plane do not overlap. When phrel is set to zero, ports in different planes may even overlap if the spatial profile of the entering wave is the same in the overlap region.

Restrictions:

1. No more than ten VOLTAGE commands may be specified.

2. If a function is specified for afscale, it must be defined prior to being referenced.

3. Time-domain only.

4. All specified voltage boundaries must be conformed to the same axis and extreme caution should be taken if they are not in the same plane.

5. The time step δt, must satisfy $\delta t < \frac{\delta x}{c \cdot phrel}$, where δx is the cell length normal to the boundary and c is the speed of light.

See Also:

FUNCTION
POISSON
LOOKBACK
PRESET

Examples:

See examples in PRESET.

X1GRID

Function: Generates a spatial grid in the x_1-coordinate.

Format:

X1GRID UNIFORM ixmax ixstart xstart dxunif ;
X1GRID EXPLICIT ixmax ixstart [xfgrid xhgrid,...] ;
X1GRID FUNCTION ixmax ixstart xstart [ncells dxstart dxtotal,...] ;

Arguments:

ixmax	-	index of the final full-grid point (integer).
ixstart	-	index of initial full-grid point (integer).
xstart	-	value of initial full-grid point (m).
dxunif	-	size of uniform grid spacing (m).
xfgrid	-	full-grid point value (m).
xhgrid	-	half-grid point value (m).
ncells	-	number of cells in region (integer).
dxstart	-	size of first cell in region (m).
dxtotal	-	total distance in region (m).

Description:

All forms of the X1GRID command generate a spatial grid in the x_1-coordinate. The spatial grid index i extends over the range,

$$1 \leq i \leq ixmax,$$

where ixmax is chosen by the user to meet simulation requirements (subject to code limitations). The spatial grid must be monotonically increasing. If the grid is specified incompletely (ixstart > 1), the code will complete the grid by symmetry. There are three options for the X1GRID command to specify a spatial grid, allowing the user much freedom in choosing a nonuniform spatial grid of the desired form.

The UNIFORM option results in exactly equally-spaced, full-grid points given by

$$x_i^f = x_1^f + (i-1)\ \delta x,$$

where δx = dxunif. Although this can also be achieved via the FUNCTION option using d = 1 δ_1^f, the UNIFORM option is the simplest way to obtain a uniform grid and avoids roundoff errors in grid spacing.

The EXPLICIT option requires the user to specify (explicitly) discrete values of pairs of full-grid and half-grid points (x_1^f, x_1^h). This way, arbitrary, nonuniform grid-spacing can be achieved.

The FUNCTION option is a simple quadratic functional prescription for nonuniform spacing. The ith full-grid point x_i^f in a particular spatial region is given by the formula,

$$x_i^f = x_1^f + (i - 1)\frac{(I^2\delta x_1^f - d)}{I(I - 1)} + (i - 1)^2\frac{(d - I\delta x_1^f)}{I(I - 1)},$$

where x_0^f = xstart, I = ncells, δx_1^f = dxstart, and d = dxtotal as specified in the input data sequence. This means that within that spatial region, the spatial grid step δx^f is allowed to increase or decrease uniformly by an amount ε, that is,

$$\delta x_{i+1}^f = \delta x_i^f \pm \varepsilon,$$

where

$$\varepsilon = \frac{2(d - I\delta x_1^f)}{I(I - 1)}.$$

A number of separate regions (subject to code constraints) can be specified for a single spatial coordinate, allowing the spatial grid that varies with different regions of the simulation.

Restrictions:

Only one X1GRID command is used to generate the entire x_1 grid.

See Also:

SYSTEM
X2GRID
X3GRID

X2GRID

Function: Generates a spatial grid in the x_2-coordinate.

Format:

Exactly the same as X1GRID.

Arguments:

Exactly the same as X1GRID, except that physical units may be either m or rad, depending upon the coordinate system.

Description:

See X1GRID.

See Also:

SYSTEM
X1GRID
X3GRID

Function: Generates a spatial grid in the x_3-coordinate.

Format:

Exactly the same as X1GRID.

Arguments:

Exactly the same as X1GRID, except that physical units may be either m or rad, depending upon the coordinate system.

Description:

See X1GRID.

See Also:

SYSTEM
X1GRID
X2GRID

Function: Preserves a command in the input file without executing it.

Format:

Z command ;

Arguments:

command - command name plus arguments (arbitrary).

Description:

The Z command is a command requesting that a line or lines of input be ignored. It is typically used to preserve a command in the input file without executing it.

The difference between the C command and the COMMENT command is that the C command does not echo in the output. The difference between the C command and the Z command is that the Z command writes the message "Command Ignored" in the output file.

See Also:

C
COMMENT.

Example:

The command,

```
Z TITLE "This is a title" ; ,
```

will preserve TITLE in the input file. A message "command ignored" will be written in the output file. Then, the TITLE command will be ignored by the code, and no text information will be processed.

SECTION 6

TEST CASES

To give specific examples of input and output for actual problems we have constructed two test cases, one frequency-domain and one time-domain. These problems and the simulation parameters are given to illustrate use of the commands. Other examples may be obtained from a variety of reports available from Mission Research Corporation.

6.1 TUNABLE KLYSTRODE CAVITY

This example is based on an actual study using the frequency-domain algorithm to determine the lowest frequency in a klystrode cavity as a function of a tunable element (a plunger) in the cavity. The cavity is cylindrical, which would allow the use of a two-dimensional code, except for the tuning element. Thus, the absolute frequency of the base cavity can be determined in two dimensions, and the function of SOS is to analyze the effect of the plunger on the frequency. The actual study was run on a VAX using approximately 27,000 cells and a series of plunger depths. Each run tested a different plunger depth and ran in approximately four hours of CPU. The example shown here was a modified run to test SOS for large problems on the CRAY. It used 256,000 cells and found 2 modes in 5 minutes of CPU. Since the input for SOS is independent of the computer used, this example applies both to VAX and CRAY. The input deck for this example is listed in Figure 6.1. A schematic of the cavity is shown in Figure 6.2. The schematic gives the physical dimensions of the cavity.

We use cartesian coordinates for the simulation for two reasons:

1. Constancy of the plunger shape as a function of depth is important.

2. In cylindrical coordinates, $r = 0$ in the cavity can slow convergence.

```
TITLE "Klystrode Cavity - plunger at 2.0 inches" ;
DEFINE  NCELLX 80 ;
DEFINE  NCELLY 80 ;
DEFINE  NCELLZ 40 ;
DEFINE "NGRDX1 = NCELLX + 1" ;
DEFINE "NGRDY1 = NCELLY + 1" ;
DEFINE "NGRDZ1 = NCELLZ + 1" ;

CONVERT CAVITYRAD  6.75  INCHES ;
CONVERT CAVITYLEN  8.5   INCHES ;
CONVERT CYLINRAD    .75  INCHES ;
CONVERT CYLOUTRAD   .875 INCHES ;
CONVERT CYLLEN     3.0   INCHES ;
CONVERT PLUNRAD    1.25  INCHES ;
CONVERT PLUNDEP    2.0   INCHES ;
CONVERT PLUNCENX  11.5   INCHES ;
CONVERT PLUNCENZ   3.25  INCHES ;
CONVERT CONEORAD1  1.15  INCHES ;
CONVERT CONEORAD2  5.25  INCHES ;
CONVERT CONEOCEN1  4.0   INCHES ;
CONVERT CONEOCEN2  8.1   INCHES ;
CONVERT CONEIRAD2  4.85  INCHES ;
CONVERT CONEICEN1  4.354 INCHES ;
CONVERT CONEICEN2  8.0   INCHES ;

DEFINE "HOLDEP = CONEICEN1 - CONEOCEN1" ;
DEFINE "EDGEDEP = CAVITYLEN - CONEOCEN2" ;

DEFINE "XCELLSZ = CAVITYRAD * 2 / NCELLX" ;
DEFINE "YCELLSZ = CAVITYRAD * 2 / NCELLY" ;
DEFINE "ZCELLSZ = CAVITYLEN / NCELLZ" ;
DEFINE "CEN = CAVITYRAD" ;
DEFINE "CAVLEFT = CAVITYLEN - CONEOCEN1" ;


X1GRID UNIFORM NGRDX1 1 0.0 XCELLSZ ;
X2GRID UNIFORM NGRDY1 1 0.0 YCELLSZ ;
DEFINE "HLFNZ = NCELLZ / 2" ;
X3GRID FUNCTION NGRDZ1 1 0.0 HLFNZ .0072 CONEOCEN1
         HLFNZ ZCELLSZ   CAVLEFT ;

STRUCTURE SOLIDFILL WALL ;
STRUCTURE CONFORMAL WALLS SOLID 1 NGRDX1 1 NGRDY1 0 1 ;
DEFINE "NGRDZ2 = NGRDZ1 + 1" ;
STRUCTURE CONFORMAL WALLS SOLID 1 NGRDX1 1 NGRDY1 NGRDZ1 NGRDZ2 ;
STRUCTURE CYLINDER CAVITY VOID CARTESIAN CEN CEN 0  3 0  1 0
     CAVITYRAD CAVITYLEN ;
```

Figure 6.1. Input deck.

```
STRUCTURE CYLINDER FERRULE SOLID CARTESIAN CEN CEN 0  3 0  1 0
     CYLOUTRAD CYLLEN ;
STRUCTURE CYLINDER FERRULE VOID CARTESIAN CEN CEN 0  3 0  1 0
     CYLINRAD CYLLEN ;
STRUCTURE CYLINDER PLUNGER SOLID CARTESIAN PLUNCENX CEN PLUNCENZ
     1 0  2 0  PLUNRAD PLUNDEP ;
STRUCTURE ELLIPTICAL CONE  SOLID CARTESIAN CEN CEN CONEOCEN1
     CEN CEN CONEOCEN2  1 0 0 CONEORAD1 CONEORAD1
     1 0 0 CONEORAD2 CONEORAD2 0 360 0 360 ;
STRUCTURE ELLIPTICAL CONE VOID CARTESIAN CEN CEN CONEICEN1
     CEN CEN CONEICEN2   1 0 0 CONEORAD1 CONEORAD1
     1 0 0 CONEIRAD2 CONEIRAD2 0 360 0 360 ;
STRUCTURE CYLINDER HOLE VOID CARTESIAN  CEN CEN CONEOCEN1
     3 0 1 0 CONEORAD1  HOLDEP ;
STRUCTURE CYLINDER EDGE1 SOLID CARTESIAN  CEN CEN CONEOCEN2
     3 0 1 0 CONEORAD2  EDGEDEP ;
STRUCTURE CYLINDER EDGE2 VOID CARTESIAN  CEN CEN CONEOCEN2
    3 0 1 0 CONEIRAD2   EDGEDEP ;

FREQUENCY 2 2.E9 .001 5000 ;
DIAGNOSE SPACING 1 0 0 ;
DIAGNOSE FREQUENCY 1 0 10 ;
DIAGNOSE FREQDEBUG 1 0 10 ;
GRAPHICS SYSTEM ;
SHOW MODEL 2 40 ;

START ;
STOP ;
```

Figure 6.1. Input deck (continued).

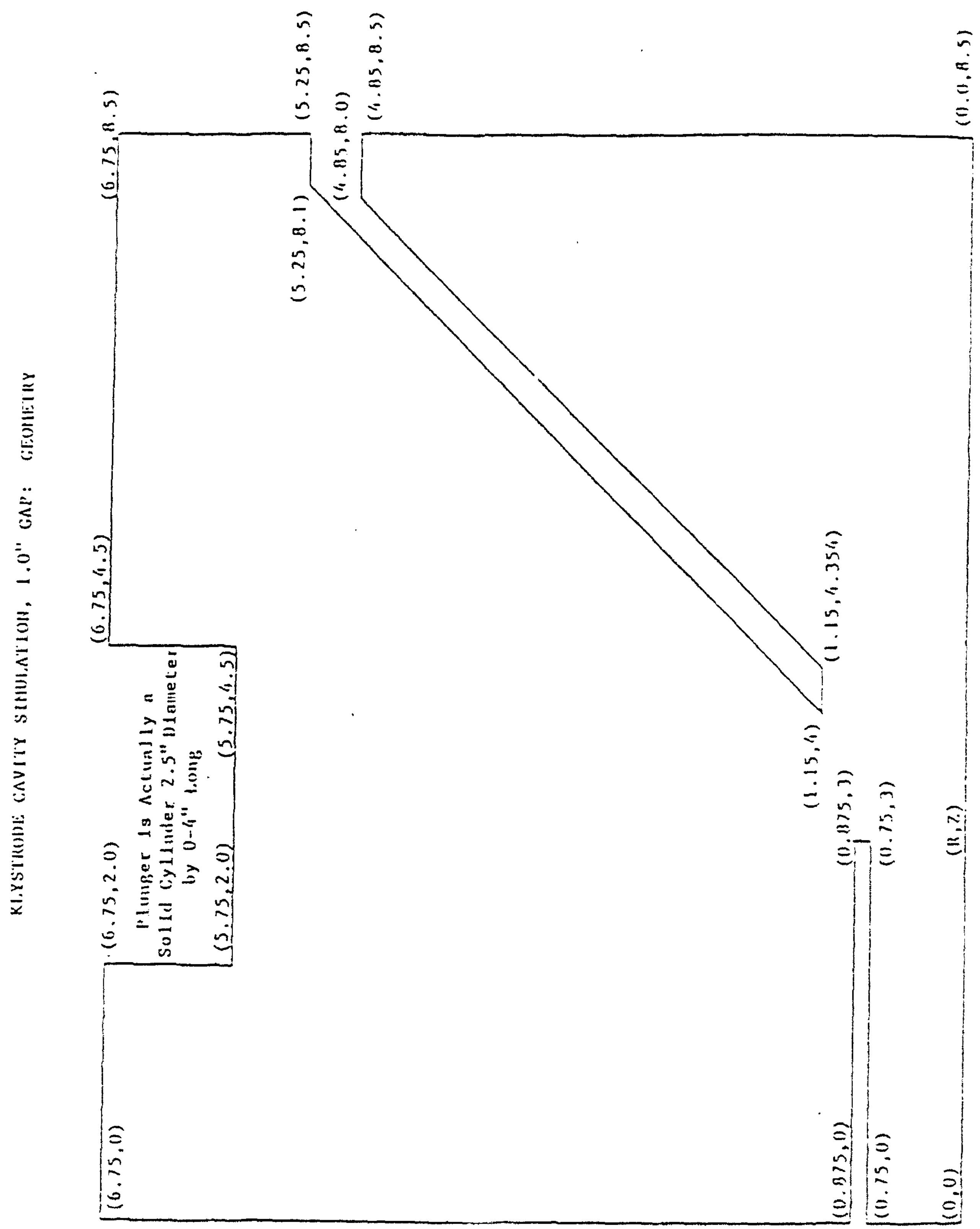


Figure 6.2. Klystrode geometry.

6.1.1 Title Command and Comments

We start off the input deck with the TITLE command, designating the name of this run. This can be followed by a COMMENT line describing the run. These commands are

```
TITLE "KLYSTRODE CAVITY - PLUNGER AT TWO INCHES";
COMMENT "THIS SIMULATION WAS PUT TOGETHER BY GARY
WARREN AND KEVIN HEANEY IN JULY OF 1988 TO TEST AND
DEMONSTRATE THE PROGRAM SOS.";
```

Note the use of the quotes to delimit alphanumeric text. The comment will be written in the processed input data in the printed output. The title will be imprinted on all of the various forms of output.

6.1.2 Unit Conversions

The geometry of the simulation given in Figure 6.2 has all the parameters specified in inches. The code works entirely in mks units, so these lengths must be converted before the run. A series of CONVERT commands converts these numbers to meters and assigns them to variables that are used later in the input deck. For example, the command

```
CONVERT PLUNDEP  2 INCHES ;
```

converts 2 inches to .0508 meters and stores this in the variable PLUNDEP. This variable is used to later specify the plunger depth in the STRUCTURE command that generates the plunger. In a series of runs for varying plunger depth, only this command line needs to be changed.

6.1.3 Grid Definition

The grid in this example is uniform in the X1 and X2 directions and variable along the X3 direction, the axis of the hollow cylinder. The number of cells and grid points along X1 are defined at the beginning of the input deck with the commands:

```
DEFINE NCELLX 80 ;
DEFINE "NGRDX1" NCELLX + 1'' ;
```

Those commands set NCELLX = 80 and NGRDX1 = 81. The grid spacing is then calculated:

DEFINE XCELLSZ CAVITYRAD 2 MULTIPLY NCELLX DIVIDE /

This command calculates the diameter of the cavity and divides this by the number of cells and then assigns that number to the variable XCELLSZ. The number of cells and spacing in the X2 and X3 directions are defined in the same way. Next the X1 grid is defined:

X1GRID UNIFORM NGRDX1 1 0.0 XCELLSZ /

Here we have chosen a uniform grid, with NGRDX1 grid points separated by XCELLSZ. The first grid point in the simulation is defined as grid point #1, and it is at the origin. If we wanted the center of the simulation to be 0.0, we could have chosen the first grid point to be 40 (NCELLX/2). By using the DEFINE command and variables in the place of numbers, we have set up the input deck so that if the number of cells needed to be changed, this could be done simply by changing the number in the first DEFINE statement. The X3 axis is defined so that the edge of the hollow cylinder and the edge of the hollow cone would both be exactly on a grid point. This is done because the system is sensitive to the geometry in the region between the cone and the cylinder. To do this, the cavity was divided into two parts, each with HLFNZ (NCELLZ/2 = 20) cells. The position of the split was chosen to be the edge of the cone, and the size of the first grid cell was varied so that the grid lined up with the edge of the cylinder. (See the X1GRID FUNCTION command in Chapter 5 for further explanation of the FUNCTION option.) The rest of the grid is set up to be uniform. The resulting command is:

```
X3GRID FUNCTION NGRDZ1 1 0.0     HLFNZ     .0072          CONEOCEN1
                                                   HLFNZ        ZCELLSZ
CAVLEFT ;
```

6.1.4 Structure Generation

The geometry in this example has a hollow cylinder and a hollow cone lined up along the same axis and a solid cylinder, the plunger, perpendicular to the axis. These three structures are enclosed in a cylindrical cavity. The first few STRUCTURE commands generate the hollow cavity.

STRUCTURE SOLIDFILL WALL ;

.

.

```
STRUCTURE CYLINDER CAVITY VOID CARTESIAN  CEN CEN 0  3 0
            1 0 CAVITYRAD CAVITYLEN ;
```

The SOLIDFILL option fills the entire grid with conductor. Two subsequent STRUCTURE commands close off the cavity using virtual cells. Then the STRUCTURE CYLINDER command with afill = VOID hollows out the cavity. The bottom of the cylinder is at the coordinates (CEN CEN 0). The cylinder is aligned with the X3 axis, and its radius and height are CAVITYRAD and CAVITYLEN respectively.

The hollow cylinder is created using two STRUCTURE CYLINDER commands with concentric cylinders of different radii. The outer cylinder is solid, and the inner cylinder is generated with afill = VOID. The cone is generated using a STRUCTURE ELLIPTICAL command with the two ellipses being circles. The cone is hollowed out using a slightly smaller concentric cone with afill = VOID. The outer cone command is:

```
STRUCTURE ELLIPTICAL CONE SOLID
  CARTESIAN  CEN  CEN  CONEOCEN1  CEN  CEN
  CONEOCEN2  1 0 0  CONEORAD1 CONEORAD1 1 0 0 CONEORAD2
  CONEORAD2  0 360  0 360 ;
```

The two ellipses are centered in the X1, X2 plane and at points CONEOCEN1, CONEOCEN2 on the X3 axis. Their primary axes point in the (1 0 0) direction. They are both circles with radii CONEORAD1 and CONEORAD2. These circles are not wedges, so the initial and final angles are specified as 0 and 360.

6.1.5 Frequency Command

The FREQUENCY command specifies that SOS is to do a frequency-domain run,

```
FREQUENCY 2 2.E9.001  5000 ;
```

In this run we are looking for the first two modes of the cavity. The maximum frequency is specified as 2 GHz. The fractional error is .001, and the maximum number of iterations is 5000. The code takes longer to run if the maximum frequency is too high. It therefore helps to know the order of magnitude of the frequencies sought.

6.1.6 Diagnostics and Graphics

The DIAGNOSE SPACING command is used to look at the grid points to check that the cylinder is lined up. This command prints the positions of all the full- and half-grid points in the output. The DIAGNOSE FREQUENCY command has information about the frequency and power shifts dumped to the output file. The command

SHOW MODEL 2 40 ;

takes a slice in the center of the X2 axis. This plane gives us the same view of the structure as the drawing with which we started (Figure 6.2). The plot resulting from this command is given in Figure 6.3.

The grid can also be plotted using a SHOW GRID command. This command creates a plot of the grid lines in a plane. It is possible to get a three-dimensional perspective plot of the conductors in the simulation. The VIEW command will generate this plot. The view of the structures is given in Figure 6.4 and was generated separately because the cylindrical cavity gets in the way of viewing the other structures. The command used to generate this plot is:

```
VIEW PLAIN .22 .4 .1 .22 CEN .1 55 X3ALIGN
        NCELLX 1 NCELLY 1 NCELLZ ;
```

The eyepoint was taken at the point (0.22,0.4,0.1). The viewpoint was the center of the simulation in the x_2 and x_3 axes. The angle of view can be used as a lens. Small angles will magnify the image (also distorting it), and large angles will reduce the size of the image like a fish-eye lens. In this case the angle was chosen to be 55 degrees. The volume of interest was the whole volume. (Again, this is without the cylindrical cavity to block our view.)

The shape of the modes can be determined by looking at vector plots of the electric and magnetic fields. No vector plots were made in this run. Several examples of vector plots are given in the second example (see Figure 6.11).

6.1.7 Results

The output file for this run is given in Figure 6.5. The arguments of each command are listed. The data from the DIAGNOSE SPACING command is listed as well as the output from the DIAGNOSE FREQUENCY command. After these numbers the final frequency results of the run are given. The first two frequencies were found after 4996 iterations. The first

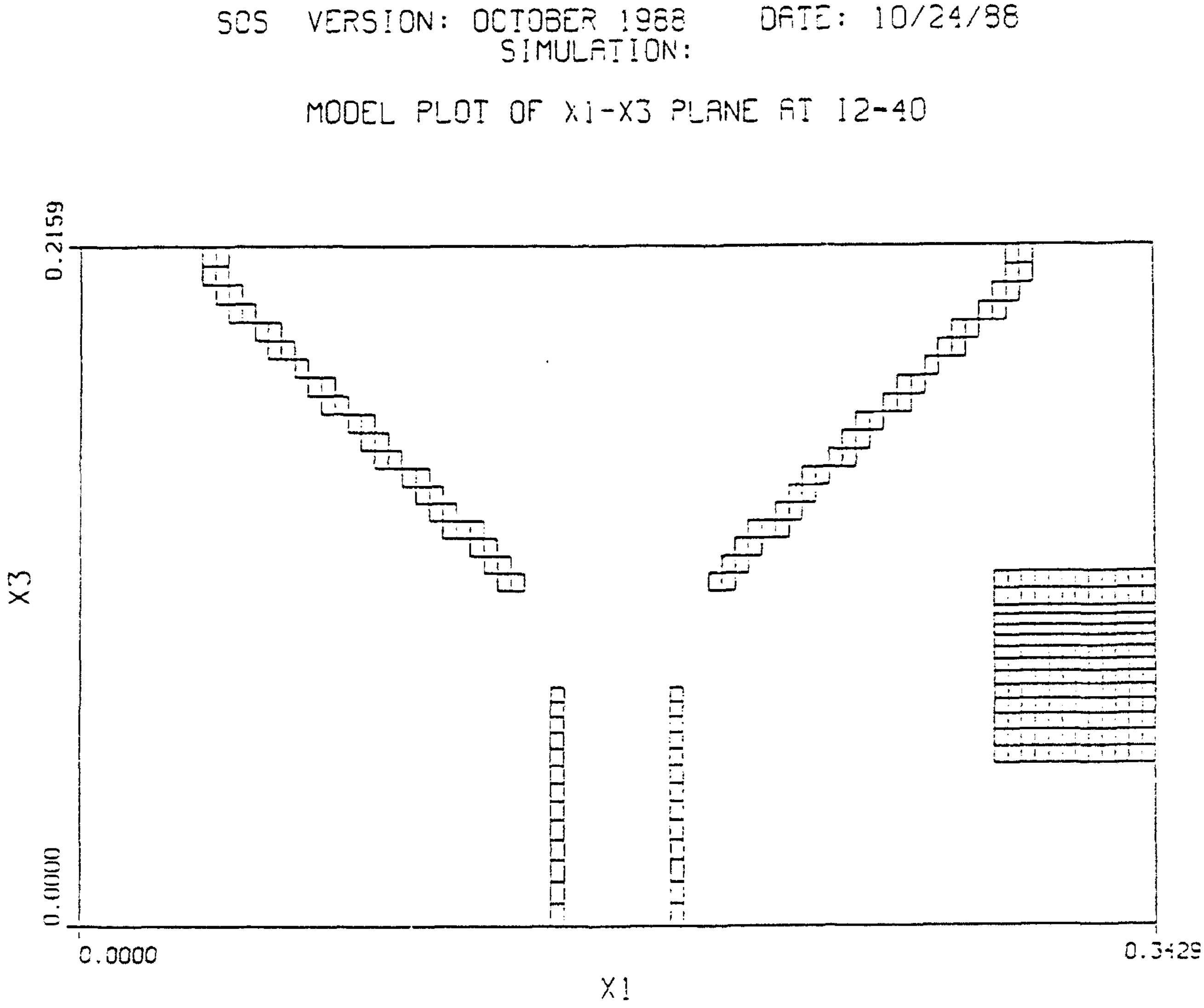


Figure 6.3. Model plot of geometry.

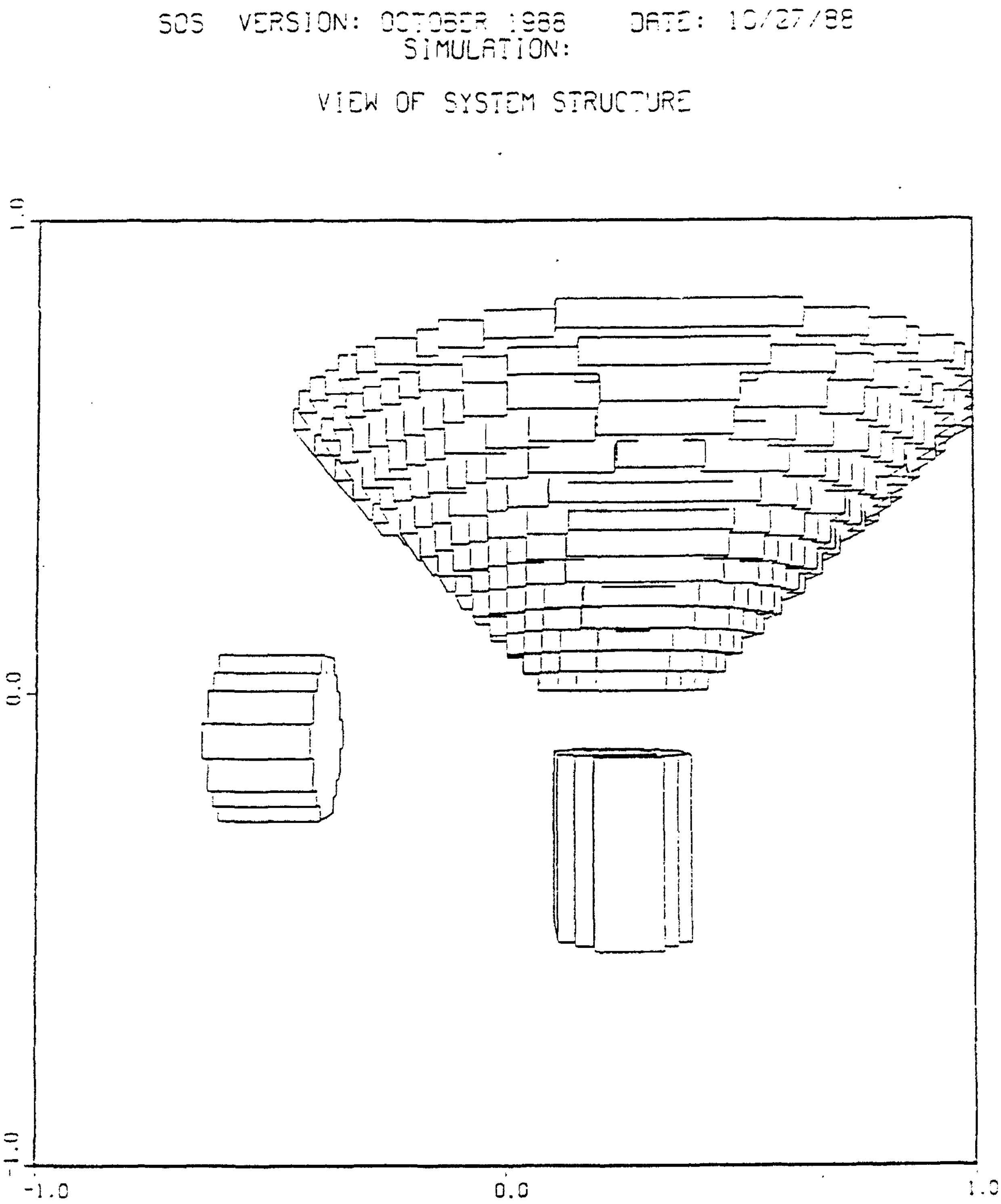


Figure 6.4. **3-D perspective plot. (Produced in a separate run).**

```
1program sos - version august 1990
1copyright 1988 mission research corporation

 begin processing input data.

 the command is: title

             simulation title =
             klystrode cavity - plunger at 2.0 inches

 the command is: define

             expression =
             ncellx=80

             result:   ncellx = 80

 the command is: define

             expression =
             ncelly=80

             result:   ncelly = 80

 the command is: define

             expression =
             ncellz=40

             result:   ncellz = 40

 the command is: define

             expression =
             ngrdx1=ncellx+1

             result:   ngrdx1 = 81

 the command is: define

             expression =
             ngrdy1=ncelly+1

             result:   ngrdy1 = 81

 the command is: define

             expression =
             ngrdz1=ncellz+1

             result:   ngrdz1 = 41

 the command is: convert

             variable name = cavityrad
             input value =    6.750e+00
             input units = inches

             result:   cavityrad = 1.714500e-01

 the command is: convert
```

Figure 6.5.   First 70 columns of processed output file.

```
          variable name = cavitylen
          input value =    8.500e+00
          input units = inches

          result:    cavitylen = 2.159000e-01
the command is: convert

          variable name = cylinrad
          input value =    7.500e-01
          input units = inches

          result:    cylinrad = 1.905000e-02
the command is: convert

          variable name = cyloutrad
          input value =    8.750e-01
          input units = inches

          result:    cyloutrad = 2.222500e-02
the command is: convert

          variable name = cyllen
          input value =    3.000e+00
          input units = inches

          result:    cyllen = 7.620000e-02
the command is: convert

          variable name = plunrad
          input value =    1.250e+00
          input units = inches

          result:    plunrad = 3.175000e-02
the command is: convert

          variable name = plundep
          input value =    2.000e+00
          input units = inches

          result:    plundep = 5.080000e-02
the command is: convert

          variable name = pluncenx
          input value =    1.150e+01
          input units = inches

          result:    pluncenx = 2.921000e-01
the command is: convert

          variable name = pluncenz
          input value =    3.250e+00
          input units = inches

          result:    pluncenz = 8.255000e-02
```

Figure 6.5. First 70 columns of processed output file (continued).

```
the command is: convert

          variable name = coneorad1
          input value =    1.150e+00
          input units = inches

          result:   coneorad1 = 2.921000e-02

the command is: convert

          variable name = coneorad2
          input value =    5.250e+00
          input units = inches

          result:   coneorad2 = 1.333500e-01

the command is: convert

          variable name = coneocen1
          input value =    4.000e+00
          input units = inches

          result:   coneocen1 = 1.016000e-01

the command is: convert

          variable name = coneocen2
          input value =    8.100e+00
          input units = inches

          result:   coneocen2 = 2.057400e-01

the command is: convert

          variable name = coneirad2
          input value =    4.850e+00
          input units = inches

          result:   coneirad2 = 1.231900e-01

the command is: convert

          variable name = coneicen1
          input value =    4.354e+00
          input units = inches

          result:   coneicen1 = 1.105916e-01

the command is: convert

          variable name = coneicen2
          input value =    8.000e+00
          input units = inches

          result:   coneicen2 = 2.032000e-01

the command is: define

          expression =
          holdep=coneicen1-coneocen1
```

Figure 6.5. First 70 columns of processed output file (continued).

```
        result:   holdep = 8.991600e-03

the command is: define

        expression =
        edgedep=cavitylen-coneocen2

        result:   edgedep = 1.016000e-02

the command is: define

        expression =
        xcellsz=cavityrad*2/ncellx

        result:   xcellsz = 4.286250e-03

the command is: define

        expression =
        ycellsz=cavityrad*2/ncelly

        result:   ycellsz = 4.286250e-03

the command is: define

        expression =
        zcellsz=cavitylen/ncellz

        result:   zcellsz = 5.397500e-03

the command is: define

        expression =
        cen=cavityrad

        result:   cen = 1.714500e-01

the command is: define

        expression =
        cavleft=cavitylen-coneocen1

        result:   cavleft = 1.143000e-01

the command is: x1grid

        xngrid generation option = uniform
        total number of grid points =     81
        first full-grid point index =      1
        first grid point value =    0.
        size of uniform grid space (m) =    4.286e-03

the command is: x2grid

        xngrid generation option = uniform
        total number of grid points =     81
        first full-grid point index =      1
        first grid point value =    0.
        size of uniform grid space (m) =    4.286e-03

the command is: define
```

Figure 6.5. First 70 columns of processed output file (continued).

```
        expression =
        hlfnz=ncellz/2

        result:    hlfnz = 2.000000e+01

the command is: x3grid

        xngrid generation option = function
        total number of grid points =     41
        first full-grid point index =      1
        first grid point value =    0.

        number of grid spaces in the region =     20
        size of first grid in the region (m) =   2.200e-03
        total distance covered by region (m) =   1.016e-01

        number of grid spaces in the region =     20
        size of first grid in the region (m) =   5.397e-03
        total distance covered by region (m) =   1.143e-01

the command is: structure

        structural element type = solidfill
        structural element name = wall


the command is: structure

        structural element type = conformal
        structural element name = walls
        inversion option = solid
        initial and final x1 indices =     1       81
        initial and final x2 indices =     1       81
        initial and final x3 indices =     0        1


the command is: define

        expression =
        ngrdz2=ngrdz1+1

        result:    ngrdz2 = 42

the command is: structure

        structural element type = conformal
        structural element name = walls
        inversion option = solid
        initial and final x1 indices =     1       81
        initial and final x2 indices =     1       81
        initial and final x3 indices =    41       42


the command is: structure

        structural element type = cylinder
        structural element name = cavity
        inversion option = void
        coordinate system for structure generation = cartesian
        translation coordinates =
                  1.715e-01     1.715e-01     0.
        polar angle axis = x3
```

Figure 6.5. First 70 columns of processed output file (continued).

```
          polar angle (degrees) - -   0.
          azimuthal angle axis - x1
          azi angle of rotation -   0.
          cylinder radius -   1.715e-01
          cylinder height -   2.159e-01

 the command is: structure

          structural element type - cylinder
          structural element name - ferrule
          inversion option - solid
          coordinate system for structure generation - cartesian
          translation coordinates -
                  1.715e-01     1.715e-01     0.
          polar angle axis - x3
          polar angle (degrees) - -   0.
          azimuthal angle axis - x1
          azi angle of rotation -   0.
          cylinder radius -   2.222e-02
          cylinder height -   7.620e-02

 the command is: structure

          structural element type - cylinder
          structural element name - ferrule
          inversion option - void
          coordinate system for structure generation - cartesian
          translation coordinates -
                  1.715e-01     1.715e-01     0.
          polar angle axis - x3
          polar angle (degrees) - -   0.
          azimuthal angle axis - x1
          azi angle of rotation -   0.
          cylinder radius -   1.905e-02
          cylinder height -   7.620e-02

 the command is: structure

          structural element type - cylinder
          structural element name - plunger
          inversion option - solid
          coordinate system for structure generation - cartesian
          translation coordinates -
                  2.921e-01     1.715e-01     8.255e-02
          polar angle axis - x1
          polar angle (degrees) - -   0.
          azimuthal angle axis - x2
          azi angle of rotation -   0.
          cylinder radius -   3.175e-02
          cylinder height -   5.080e-02

 the command is: structure

          structural element type - elliptical
          structural element name - cone
          inversion option - solid
          coordinate system for structure generation - cartesian
          translation coordinates -
```

Figure 6.5. First 70 columns of processed output file (continued).

```
                 1.715e-01     1.715e-01     1.016e-01
         co. of center of 2nd ellipse -
                 1.715e-01     1.715e-01     2.057e-01
         co. for primary axis of 1st ell. -
                 1.000e+00     0.            0.
         length of axes of 1st ellipse -   2.921e-02     2.921e-02
         co. for primary axis of 2nd ell. -
                 1.000e+00     0.            0.
         length of axes of 2nd ellipse -   1.334e-01     1.334e-01
         in & fi angle for section of 1st ell. -   0.            3.600e+02
         in & fi angle for section of 2nd ell. -   0.            3.600e+02

the command is: structure

         structural element type - elliptical
         structural element name - cone
         inversion option - void
         coordinate system for structure generation - cartesian
         translation coordinates -
                 1.715e-01     1.715e-01     1.106e-01
         co. of center of 2nd ellipse -
                 1.715e-01     1.715e-01     2.032e-01
         co. for primary axis of 1st ell. -
                 1.000e+00     0.            0.
         length of axes of 1st ellipse -   2.921e-02     2.921e-02
         co. for primary axis of 2nd ell. -
                 1.000e+00     0.            0.
         length of axes of 2nd ellipse -   1.232e-01     1.232e-01
         in & fi angle for section of 1st ell. -   0.            3.600e+02
         in & fi angle for section of 2nd ell. -   0.            3.600e+02

the command is: structure

         structural element type - cylinder
         structural element name - hole
         inversion option - void
         coordinate system for structure generation - cartesian
         translation coordinates -
                 1.715e-01     1.715e-01     1.016e-01
         polar angle axis - x3
         polar angle (degrees) - -   0.
         azimuthal angle axis - x1
         azi angle of rotation -   0.
         cylinder radius -   2.921e-02
         cylinder height -   8.992e-03

the command is: structure

         structural element type - cylinder
         structural element name - edge1
         inversion option - solid
         coordinate system for structure generation - cartesian
         translation coordinates -
                 1.715e-01     1.715e-01     2.057e-01
         polar angle axis - x3
         polar angle (degrees) - -   0.
         azimuthal angle axis - x1
         azi angle of rotation -   0.
         cylinder radius -   1.334e-01
```

Figure 6.5. First 70 columns of processed output file (continued).

```
        cylinder height -   1.016e-02

the command is: structure

        structural element type - cylinder
        structural element name - edge2
        inversion option - void
        coordinate system for structure generation - cartesian
        translation coordinates -
                1.715e-01      1.715e-01      2.057e-01
        polar angle axis - x3
        polar angle (degrees) - -   0.
        azimuthal angle axis - x1
        azi angle of rotation -   0.
        cylinder radius -   1.232e-01
        cylinder height -   1.016e-02

the command is: frequency

        number of resonances sought -        2
        highest frequency resonance sought -   2.000e+09
        fractional accuracy sought -   1.000e-03
        total iteration limit -  5000

the command is: diagnose

        diagnostic type - spacing
        interval,start,and stop -      1        0        0

the command is: diagnose

        diagnostic type - frequency
        interval,start,and stop -      1        0       10

the command is: diagnose

        diagnostic type - freqdebug
        interval,start,and stop -      1        0       10

the command is: graphics

        graphics output device - system

the command is: show

        display option - model
        viewing axis - x2
        axis coordinate index -     40

the command is: start

            54 commands read.
no input errors.
```

Figure 6.5. First 70 columns of processed output file (continued).

```
*** the simulation run has started ***


spacing          diagnostic

          x1 coordinate,  81 full-grid points, system =     0
          full-grid point values
             0.00000    0.00429    0.00857    0.01286    0.01714    0.02143    0.02572    0.03000    0.03429    0.03858
             0.04286    0.04715    0.05144    0.05572    0.06001    0.06429    0.06858    0.07287    0.07715    0.08144
             0.08572    0.09001    0.09430    0.09858    0.10287    0.10716    0.11144    0.11573    0.12001    0.12430
             0.12859    0.13287    0.13716    0.14145    0.14573    0.15002    0.15430    0.15859    0.16288    0.16716
             0.17145    0.17574    0.18002    0.18431    0.18859    0.19288    0.19717    0.20145    0.20574    0.21003
             0.21431    0.21860    0.22288    0.22717    0.23146    0.23574    0.24003    0.24432    0.24860    0.25289
             0.25717    0.26146    0.26575    0.27003    0.27432    0.27861    0.28289    0.28718    0.29146    0.29575
             0.30004    0.30432    0.30861    0.31290    0.31718    0.32147    0.32575    0.33004    0.33433    0.33861
             0.34290
          half-grid point values
             0.00214    0.00643    0.01072    0.01500    0.01929    0.02357    0.02786    0.03215    0.03643    0.04072
             0.04501    0.04929    0.05358    0.05786    0.06215    0.06644    0.07072    0.07501    0.07930    0.08358
             0.08787    0.09215    0.09644    0.10073    0.10501    0.10930    0.11359    0.11787    0.12216    0.12644
             0.13073    0.13502    0.13930    0.14359    0.14788    0.15216    0.15645    0.16073    0.16502    0.16931
             0.17359    0.17788    0.18217    0.18645    0.19074    0.19502    0.19931    0.20360    0.20788    0.21217
             0.21646    0.22074    0.22503    0.22931    0.23360    0.23789    0.24217    0.24646    0.25075    0.25503
             0.25932    0.26360    0.26789    0.27218    0.27646    0.28075    0.28504    0.28932    0.29361    0.29789
             0.30218    0.30647    0.31075    0.31504    0.31933    0.32361    0.32790    0.33218    0.33647    0.34076
             0.34504
          x2 coordinate,  81 full-grid points, system =     0
          full-grid point values
             0.00000    0.00429    0.00857    0.01286    0.01714    0.02143    0.02572    0.03000    0.03429    0.03858
             0.04286    0.04715    0.05144    0.05572    0.06001    0.06429    0.06858    0.07287    0.07715    0.08144
             0.08572    0.09001    0.09430    0.09858    0.10287    0.10716    0.11144    0.11573    0.12001    0.12430
             0.12859    0.13287    0.13716    0.14145    0.14573    0.15002    0.15430    0.15859    0.16288    0.16716
             0.17145    0.17574    0.18002    0.18431    0.18859    0.19288    0.19717    0.20145    0.20574    0.21003
             0.21431    0.21860    0.22288    0.22717    0.23146    0.23574    0.24003    0.24432    0.24860    0.25289
             0.25717    0.26146    0.26575    0.27003    0.27432    0.27861    0.28289    0.28718    0.29146    0.29575
             0.30004    0.30432    0.30861    0.31290    0.31718    0.32147    0.32575    0.33004    0.33433    0.33861
             0.34290
          half-grid point values
             0.00214    0.00643    0.01072    0.01500    0.01929    0.02357    0.02786    0.03215    0.03643    0.04072
             0.04501    0.04929    0.05358    0.05786    0.06215    0.06644    0.07072    0.07501    0.07930    0.08358
             0.08787    0.09215    0.09644    0.10073    0.10501    0.10930    0.11359    0.11787    0.12216    0.12644
             0.13073    0.13502    0.13930    0.14359    0.14788    0.15216    0.15645    0.16073    0.16502    0.16931
             0.17359    0.17788    0.18217    0.18645    0.19074    0.19502    0.19931    0.20360    0.20788    0.21217
             0.21646    0.22074    0.22503    0.22931    0.23360    0.23789    0.24217    0.24646    0.25075    0.25503
             0.25932    0.26360    0.26789    0.27218    0.27646    0.28075    0.28504    0.28932    0.29361    0.29789
             0.30218    0.30647    0.31075    0.31504    0.31933    0.32361    0.32790    0.33218    0.33647    0.34076
             0.34504
          x3 coordinate,  41 full-grid points, system =     0
          full-grid point values
             0.00000    0.00720    0.01418    0.02093    0.02746    0.03377    0.03985    0.04571    0.05125    0.05677
             0.06196    0.06693    0.07167    0.07619    0.08049    0.08457    0.08842    0.09205    0.09546    0.09864
             0.10160    0.10700    0.11243    0.11789    0.12339    0.12892    0.13449    0.14008    0.14572    0.15138
             0.15708    0.16281    0.16858    0.17437    0.18021    0.18607    0.19197    0.19790    0.20387    0.20967
             0.21590
          half-grid point values
             0.00363    0.01072    0.01758    0.02422    0.03064    0.03684    0.04281    0.04856    0.05409    0.05939
             0.06447    0.06933    0.07396    0.07837    0.08256    0.08652    0.09026    0.09378    0.09708    0.10015
             0.10429    0.10971    0.11516    0.12064    0.12615    0.13170    0.13728    0.14290    0.14854    0.15423
             0.15994    0.16569    0.17147    0.17729    0.18313    0.18902    0.19493    0.20088    0.20686    0.21288
```

Figure 6.5. First 70 columns of processed output file (continued).

```
            0.21892
maximum frequency is:    0.420963e+11
last estimate was:    0.421004e+11

itert =        0    freqt =   0.             wdesf =   1.0000e+00    ishift =       0
nshftn =      20    nshfts =        0    ndotmn =       4    ndot =      40    ifull = -32004


start of search for mode =      1
fractional accuracy sought =         5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.             4.0000e-18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     9.4463e+18   1.7694e+21   3.1615e+20   9.5867e+20
power-shifts =     1.3680e+20   1.6420e+21   8.2016e+20   1.4627e+21
power-shifts =     3.1114e+19   1.7477e+21   4.2821e+20   1.0955e+21
power-shifts =     2.1821e+20   1.5606e+21   6.8335e+20   1.3506e+21
power-shifts =     7.3916e+19   1.7049e+21   5.5162e+20   1.2272e+21
power-shifts =
     40  2.611121e+10  2.611121e+10  1.000000e+00  1.000000e+00  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =       40    freqt =   1.3057e+01    wdesf =   1.0000e+00    ishift =       0
nshftn =       4    nshfts =       20    ndotmn =       4    ndot =       8    ifull = -32004


start of search for mode =      1
fractional accuracy sought =         5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.             4.0000e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     7.6321e+20   1.7322e+21   1.0470e+21   1.4484e+21
power-shifts =
     48  3.352748e+10  3.352748e+10  5.000000e-01  5.000000e-01  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =       48    freqt =   1.6764e+01    wdesf =   1.0000e+00    ishift = -32004
nshftn =       4    nshfts =        4    ndotmn =       4    ndot =       8    ifull = -32004


start of search for mode =      1
fractional accuracy sought =         5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.             4.0000e-18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     9.5406e+20   1.7397e+21   1.1842e+21   1.5096e+21
power-shifts =
     56  7.152345e+09  7.152345e+09  5.000000e-01  5.000000e-01  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =       56    freqt =   1.7641e+01    wdesf =   1.0000e+00    ishift = -32004
nshftn =       4    nshfts =        4    ndotmn =       4    ndot =       8    ifull = -32004


start of search for mode =      1
fractional accuracy sought =         5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.             4.0000e+18   4.0000e+18   4.0000e-18   1.7721e+21
power-shifts =     1.7966e+20   1.7091e+21   6.2762e+20   1.2611e+21
power-shifts =
     64  6.031860e+09  5.555866e+09  8.567414e-02  4.255511e-03  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =       64    freqt =   3.5762e+00    wdesf =   1.0000e+00    ishift = -32003
nshftn =       4    nshfts =        4    ndotmn =       4    ndot =       8    ifull = -32004


start of search for mode =      1
fractional accuracy sought =         5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.             4.0000e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     1.6599e+20   1.7086e+21   6.1780e+20   1.2567e+21
power-shifts =
```

Figure 6.5. First 70 columns of processed output file (continued).

```
     72  5.199198e-09  3.918333e+09  3.268904e-01  2.679463e-03  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =      72    freqt =   3.0159e+00    wdesf =   1.0000e+00    ishift = -32003
nshftn =      8    nshfts =       4    ndotmn =       4    ndot =      16    ifull = -32004

start of search for mode =      1
fractional accuracy sought =        5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.            4.0000e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     6.0402e+19   1.7555e+21   4.2785e+20   1.0765e+21
power-shifts =     1.8943e+20   1.6265e+21   7.3936e+20   1.3880e+21
power-shifts =
     88  3.877718e+09  3.600878e-09  7.688137e-02  3.413360e-03  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =      88    freqt =   2.5996e+00    wdesf =   1.0000e+00    ishift = -32003
nshftn =      4    nshfts =       8    ndotmn =       4    ndot =       8    ifull = -32004

start of search for mode =      1
fractional accuracy sought =        5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.            4.0000e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     1.4624e+20   1.7078e+21   6.0360e+20   1.2504e+21
power-shifts =
     96  3.596439e+09  3.596439e+09  5.000000e-01  5.975362e-04  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =      96    freqt =   1.9389e+00    wdesf =   1.0000e+00    ishift = -32004
nshftn =     12    nshfts =       4    ndotmn =       4    ndot =      24    ifull = -32004

start of search for mode =      1
fractional accuracy sought =        5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.            4.0000e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     2.7952e+19   1.7646e+21   3.5711e+20   1.0106e+21
power-shifts =     2.0144e+20   1.5911e+21   7.8196e+20   1.4294e+21
power-shifts =     8.7127e+19   1.7054e+21   5.6112e+20   1.2314e+21
power-shifts =
    120  1.790930e+09  1.668896e+09  7.312243e-02  2.749448e-03  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =     120    freqt =   1.7982e+00    wdesf =   1.0000e+00    ishift = -32003
nshftn =     24    nshfts =      12    ndotmn =       4    ndot =      48    ifull = -32004

start of search for mode =      1
fractional accuracy sought =        5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.            3.2074e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     7.7836e+18   1.7702e+21   3.0673e+20   9.4675e+20
power-shifts =     9.6965e+19   1.6810e+21   8.3124e+20   1.4713e+21
power-shifts =     2.2861e+19   1.7551e+21   3.9837e+20   1.0613e+21
power-shifts =     1.5472e+20   1.6233e+21   7.1671e+20   1.3796e+21
power-shifts =     5.2759e+19   1.7252e+21   4.9841e+20   1.1729e+21
power-shifts =     2.2504e+20   1.5529e+21   6.0513e+20   1.2796e+21
power-shifts =
    168  1.378622e+09  1.293608e+09  6.571895e-02  3.691201e-04  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =     168    freqt =   8.9546e-01    wdesf =   1.0000e+00    ishift = -32003
nshftn =     44    nshfts =      24    ndotmn =       4    ndot =     176    ifull = -32003

start of search for mode =      1
fractional accuracy sought =        5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.            1.9006e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =     5.1264e+18   1.7715e+21   2.8612e+20   9.1987e+20
```

Figure 6.5. **First 70 columns of processed output file (continued).**

```
power-shifts =    9.8044e+19   1.6786e+21   8.5679e+20   1.4905e+21
power-shifts =    9.6270e+18   1.7670e+21   3.3379e+20   9.8280e+20
power-shifts =    1.2828e+20   1.6484e+21   7.9386e+20   1.4429e+21
power-shifts =    1.8605e+19   1.7581e+21   3.8430e+20   1.0452e+21
power-shifts =    1.6239e+20   1.6143e+21   7.3141e+20   1.2924e+21
power-shifts =    3.2015e+19   1.7446e+21   4.3737e+20   1.1069e+21
power-shifts =    2.0019e+20   1.5765e+21   6.6977e+20   1.3393e+21
power-shifts =    4.9789e+19   1.7269e+21   4.9274e+20   1.1674e+21
power-shifts =    2.4151e+20   1.5352e+21   6.0924e+20   1.2839e+21
power-shifts =    7.1836e+19   1.7048e+21   5.5013e+20   1.2265e+21
power-shifts =
    344  9.761044e+08  9.761044e+08  5.000000e-01  2.675725e-04  0  0  0  0     0  0  0  0  0  0  0  0  0  0
itert =     344     freqt =   6.8931e-01     wdesf =    1.0000e+00     ishift = -32004
nshftn =     84     nshfts =      44     ndotmn =        4     ndot =     168     ifull = -32004
itert =     344     freqt =   6.8931e-01     wdesf =    1.0000e+00     ishift = -32004
nshftn =     60     nshfts =      44     ndotmn =        4     ndot =     120     ifull = -32004

start of search for mode =        1
fractional accuracy sought =        5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.          1.3786e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    4.6058e+18   1.7718e+21   2.7977e+20   9.1134e+20
power-shifts =    9.0408e+19   1.6860e+21   8.6506e+20   1.4966e+21
power-shifts =    7.0277e+18   1.7694e+21   3.1415e+20   9.5755e+20
power-shifts =    1.1142e+20   1.6650e+21   8.1885e+20   1.4622e+21
power-shifts =    1.1865e+19   1.7645e+21   3.5012e+20   1.0036e+21
power-shifts =    1.3455e+20   1.6418e+21   7.7283e+20   1.4263e+21
power-shifts =    1.9104e+19   1.7573e+21   3.8756e+20   1.0493e+21
power-shifts =    1.5976e+20   1.6166e+21   7.2712e+20   1.3888e+21
power-shifts =    2.8725e+19   1.7477e+21   4.2637e+20   1.0945e+21
power-shifts =    1.8696e+20   1.5894e+21   6.8186e+20   1.3500e+21
power-shifts =    4.0702e+19   1.7257e+21   4.6644e+20   1.1392e+21
power-shifts =    2.1608e+20   1.5603e+21   6.3716e+20   1.3100e+21
power-shifts =    5.5002e+19   1.7214e+21   5.0767e+20   1.1833e+21
power-shifts =    2.4704e+20   1.5294e+21   5.9315e+20   1.2687e+21
power-shifts =    7.1586e+19   1.7048e+21   5.4995e+20   1.2265e+21
power-shifts =
    464  8.820591e+08  7.121730e+08  2.385461e-01  1.695133e-04  0  0  0  1     1  0  0  0  0  0  0  0  0  0
itert =     464     freqt =   5.8706e-01     wdesf =    1.0000e+00     ishift = -32003
nshftn =    108     nshfts =      60     ndotmn =        4     ndot =     432     ifull = -32003

start of search for mode =        1
fractional accuracy sought =        5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.          7.7803e+17   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    4.1870e+18   1.7720e+21   2.7217e+20   9.0095e+20
power-shifts =    8.1569e+19   1.6946e+21   8.7524e+20   1.5040e+21
power-shifts =    4.9349e+18   1.7713e+21   2.9087e+20   9.2666e+20
power-shifts =    9.2436e+19   1.6838e+21   8.4954e+20   1.4853e+21
power-shifts =    6.4300e+18   1.7698e+21   3.1008e+20   9.5233e+20
power-shifts =    1.0398e+20   1.6722e+21   8.2387e+20   1.4661e+21
power-shifts =    8.6711e+18   1.7675e+21   3.2978e+20   9.7794e+20
power-shifts =    1.1618e+20   1.6600e+21   7.9825e+20   1.4464e+21
power-shifts =    1.1656e+19   1.7645e+21   3.4995e+20   1.0035e+21
power-shifts =    1.2904e+20   1.6472e+21   7.7271e+20   1.4262e+21
power-shifts =    1.5383e+19   1.7608e+21   3.7057e+20   1.0289e+21
power-shifts =    1.4254e+20   1.6337e+21   7.4727e+20   1.4056e+21
power-shifts =    1.9848e+19   1.7563e+21   3.9164e+20   1.0542e+21
power-shifts =    1.5667e+20   1.6195e+21   7.2194e+20   1.3846e+21
power-shifts =    2.5048e+19   1.7511e+21   4.1312e+20   1.0794e+21
```

Figure 6.5. First 70 columns of processed output file (continued).

```
power-shifts -     1.7141e+20    1.6048e+21    6.9676e+20    1.3631e+21
power-shifts -     3.0978e+19    1.7452e+21    4.3501e+20    1.1044e+21
power-shifts -     1.8677e+20    1.5894e+21    6.7174e+20    1.3412e+21
power-shifts -     3.7633e+19    1.7386e+21    4.5728e+20    1.1293e+21
power-shifts -     2.0272e+20    1.5735e+21    6.4690e+20    1.3189e+21

power-shifts -     4.5008e+19    1.7312e+21    4.7991e+20    1.1539e+21
power-shifts -     2.1925e+20    1.5569e+21    6.2227e+20    1.2963e+21
power-shifts -     5.2036e+19    1.7231e+21    5.0289e+20    1.1783e+21
power-shifts -     2.363[illegible]e+20    1.5399e+21    5.9786e+20    1.2733e+21
power-shifts -     6.1891e+19    1.7143e+21    5.2619e+20    1.2025e+21
power-shifts -     2.5399e+20    1.5222e+21    5.7370e+20    1.2500e+21
power-shifts -     7.1384e+19    1.7048e+21    5.4980e+20    1.2264e+21
power-shifts -
    896  7.410146e+08  7.317940e+08  1.259992e-01  6.461177e-05  0  0  0  2     1  0  0  0  0  0  0  0  0  0
itert -     896     freqt -   4.4103e-01     wdesf -    1.0000e+00     ishift - -32003
nshftn -     120     nshfts -     108    ndotmn -       4     ndot -     480     ifull - -32003

start of search for mode -      1
fractional accuracy sought -        5.9e-04
w2min,freq2,w2des,w2desf,w2max -
           0.            5.4910e+17    4.0000e+18    4.0000e+18    1.7721e+21
power-shifts -     4.1515e+18    1.7720e+21    2.7123e+20    8.9966e+20
power-shifts -     7.5865e+19    1.7003e+21    8.7652e+20    1.5049e+21
power-shifts -     4.7573e+18    1.7714e+21    2.8802e+20    9.2279e+20
power-shifts -     8.5276e+19    1.6909e+21    8.5338e+20    1.4882e+21
power-shifts -     5.9685e+18    1.7702e+21    3.0522e+20    9.4590e+20
power-shifts -     9.5241e+19    1.6809e+21    8.3027e+20    1.4710e+21
power-shifts -     7.7842e+18    1.7684e+21    3.2282e+20    9.6898e+20
power-shifts -     1.0575e+20    1.6704e+21    8.0720e+20    1.4534e+21
power-shifts -     1.0203e+19    1.7660e+21    3.4080e+20    9.9199e+20
power-shifts -     1.1679e+20    1.6594e+21    7.8418e+20    1.4354e+21
power-shifts -     1.3224e+19    1.7629e+21    3.5916e+20    1.0149e+21
power-shifts -     1.2936e+20    1.6478e+21    7.6124e+20    1.4170e+21
power-shifts -     1.6844e+19    1.7593e+21    3.7788e+20    1.0378e+21
power-shifts -     1.4045e+20    1.6357e+21    7.3838e+20    1.3983e+21
power-shifts -     2.1062e+19    1.7551e+21    3.9696e+20    1.0605e+21
power-shifts -     1.5306e+20    1.6231e+21    7.1562e+20    1.3792e+21
power-shifts -     2.5873e+19    1.7503e+21    4.1637e+20    1.0832e+21
power-shifts -     1.6617e+20    1.6100e+21    6.9299e+20    1.3598e+21
power-shifts -     3.1276e+19    1.7449e+21    4.3610e+20    1.1057e+21
power-shifts -     1.7977e+20    1.5964e+21    6.7048e+20    1.3401e+21

power-shifts -     3.7266e+19    1.7389e+21    4.5614e+20    1.1280e+21
power-shifts -     1.9386e+20    1.5823e+21    6.4813e+20    1.3200e+21
power-shifts -     4.3839e+19    1.7323e+21    4.7648e+20    1.1502e+21
power-shifts -     2.0842e+20    1.5678e+21    6.2594e+20    1.2997e+21
power-shifts -     5.0990e+19    1.7252e+21    4.9710e+20    1.1722e+21
power-shifts -     2.2345e+20    1.5527e+21    6.0393e+20    1.2791e+21
power-shifts -     5.8715e+19    1.7175e+21    5.1799e+20    1.1941e+21
power-shifts -     2.3894e+20    1.5372e+21    5.8[illegible]12e+20    1.2582e+21
power-shifts -     6.7009e+19    1.7092e+21    5.3913e+20    1.2157e+21
power-shifts -     2.5487e+20    1.5213e+21    5.6051e+20    1.2370e+21
power-shifts -
   1376  7.106532e+08  7.106532e+08  3.812574e-02  1.243927e-05  0  0  0  3     1  0  0  0  0  0  0  0  0  0
itert -    1376     freqt -   3.7051e-01     wdesf -    1.0000e+00     ishift - -32004
nshftn -     108     nshfts -     120    ndotmn -       4     ndot -     216     ifull - -32004

start of search for mode -      1
fractional accuracy sought -        5.9e-04
```

Figure 6.5. **First 70 columns of processed output file (continued).**

```
w2min,freq2,w2des,w2desf,w2max =
            0.          5.0503e+17   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    4.1870e+18   1.7720e+21   2.7217e+20   9.0095e+20
power-shifts =    8.1569e+19   1.6946e+21   8.7524e+20   1.5040e+21
power-shifts =    4.9349e+18   1.7713e+21   2.9087e+20   9.2666e+20
power-shifts =    9.2436e+19   1.6838e+21   8.4954e+20   1.4853e+21
power-shifts =    6.4300e+18   1.7698e+21   3.1008e+20   9.5233e+20
power-shifts =    1.0398e+20   1.6722e+21   8.2387e+20   1.4661e+21
power-shifts =    8.6711e+18   1.7675e+21   3.2978e+20   9.7794e+20
power-shifts =    1.1618e+20   1.6600e+21   7.9825e+20   1.4464e+21
power-shifts =    1.1656e+19   1.7645e+21   3.4995e+20   1.0035e+21
power-shifts =    1.2904e+20   1.6472e+21   7.7271e+20   1.4262e+21
power-shifts =    1.5383e+19   1.7608e+21   3.7057e+20   1.0289e+21
power-shifts =    1.4254e+20   1.6337e+21   7.4727e+20   1.4056e+21
power-shifts =    1.9848e+19   1.7563e+21   3.9164e+20   1.0542e+21
power-shifts =    1.5667e+20   1.6195e+21   7.2194e+20   1.3846e+21
power-shifts =    2.5048e+19   1.7511e+21   4.1312e+20   1.0794e+21
power-shifts =    1.7141e+20   1.6048e+21   6.9676e+20   1.3631e+21
power-shifts =    3.0978e+19   1.7452e+21   4.3501e+20   1.1044e+21
power-shifts =    1.8677e+20   1.5894e+21   6.7174e+20   1.3412e+21
power-shifts =    3.7633e+19   1.7386e+21   4.5728e+20   1.1293e+21
power-shifts =    2.0272e+20   1.5735e+21   6.4690e+20   1.3189e+21

power-shifts =    4.5008e+19   1.7312e+21   4.7991e+20   1.1539e+21
power-shifts =    2.1925e+20   1.5569e+21   6.2227e+20   1.2963e+21
power-shifts =    5.3096e+19   1.7231e+21   5.0289e+20   1.1783e+21
power-shifts =    2.3634e+20   1.5399e+21   5.9786e+20   1.2733e+21
power-shifts =    6.1891e+19   1.7143e+21   5.2619e+20   1.2025e+21
power-shifts =    2.5399e+20   1.5222e+21   5.7370e+20   1.2500e+21
power-shifts =    7.1384e+19   1.7048e+21   5.4980e+20   1.2264e+21
power-shifts =
   1592  6.130147e+08  6.130147e+08  3.504034e-01  3.647326e-05  0  0  0  4   1   0  0  0  0  0  0  0  0  0
itert =   1592      freqt =   3.5533e-01     wdesf =   1.0000e+00     ishift = -32004
nshftn =    240     nshfts =    108     ndotmn =      4     ndot =    480     ifull = -32004

start of search for mode =      1
fractional accuracy sought =          5.9e-04
w2min,freq2,w2des,w2desf,w2max =
            0.          3.7579e+17   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    4.0379e+18   1.7721e+21   2.6705e+20   8.9384e+20
power-shifts =    7.3544e+19   1.7026e+21   8.8227e+20   1.7091e+21
power-shifts =    4.1893e+18   1.7719e+21   2.7534e+20   9.0542e+20
power-shifts =    7.8112e+19   1.6980e+21   8.7070e+20   1.5008e+21
power-shifts =    4.4923e+18   1.7716e+21   2.8374e+20   9.1698e+20
power-shifts =    8.2819e+19   1.6933e+21   8.5913e+20   1.4924e+21
power-shifts =    4.9466e+18   1.7712e+21   2.9223e+20   9.2855e+20
power-shifts =    8.7664e+19   1.6885e+21   8.4757e+20   1.4839e+21
power-shifts =    5.5522e+18   1.7706e+21   3.0083e+20   9.4010e+20
power-shifts =    9.2645e+19   1.6835e+21   8.3601e+20   1.4753e+21
power-shifts =    6.3090e+18   1.7698e+21   3.0953e+20   9.5165e+20
power-shifts =    9.7764e+19   1.6784e+21   8.2447e+20   1.4666e+21
power-shifts =    7.2170e+18   1.7689e+21   3.1833e+20   9.6319e+20
power-shifts =    1.0302e+20   1.6731e+21   8.1293e+20   1.4578e+21
power-shifts =    8.2758e+18   1.7678e+21   3.2723e+20   9.7471e+20
power-shifts =    1.0841e+20   1.6677e+21   8.0141e+20   1.4489e+21
power-shifts =    9.4854e+18   1.7666e+21   3.3622e+20   9.8622e+20
power-shifts =    1.1393e+20   1.6622e+21   7.8990e+20   1.4399e+21
power-shifts =    1.0846e+19   1.7653e+21   3.4531e+20   9.9771e+20
power-shifts =    1.1958e+20   1.6565e+21   7.7841e+20   1.4308e+21

power-shifts =    1.2356e+19   1.7638e+21   3.5449e+20   1.0092e+21
```

Figure 6.5. First 70 columns of processed output file (continued).

```
power-shifts =    1.2537e+20   1.6507e+21   7.6693e+20   1.4216e+21
power-shifts =    1.4016e+19   1.7621e+21   3.6376e+20   1.0206e+21
power-shifts =    1.3128e+20   1.6448e+21   7.5548e+20   1.4124e+21
power-shifts =    1.5827e+19   1.7603e+21   3.7313e+20   1.0321e+21
power-shifts =    1.3733e+20   1.6388e+21   7.4405e+20   1.4030e+21
power-shifts =    1.7786e+19   1.7583e+21   3.8258e+20   1.0435e+21
power-shifts =    1.4351e+20   1.6326e+21   7.3265e+20   1.3935e+21
power-shifts =    1.9895e+19   1.7562e+21   3.9211e+20   1.0548e+21
power-shifts =    1.4981e+20   1.6263e+21   7.2127e+20   1.3840e+21
power-shifts =    2.2153e+19   1.7540e+21   4.0173e+20   1.0662e+21
power-shifts =    1.5624e+20   1.6199e+21   7.0992e+20   1.3744e+21
power-shifts =    2.4559e+19   1.7516e+21   4.1144e+20   1.0775e+21
power-shifts =    1.6279e+20   1.6133e+21   6.9860e+20   1.3647e+21
power-shifts =    2.7113e+19   1.7490e+21   4.2123e+20   1.0888e+21
power-shifts =    1.6947e+20   1.6066e+21   6.8731e+20   1.3549e+21
power-shifts =    2.9814e+19   1.7463e+21   4.3109e+20   1.1001e+21
power-shifts =    1.7627e+20   1.5998e+21   6.7606e+20   1.3450e+21
power-shifts =    3.2663e+19   1.7435e+21   4.4104e+20   1.1113e+21
power-shifts =    1.8320e+20   1.5929e+21   6.6485e+20   1.3351e+21

power-shifts =    3.5658e+19   1.7405e+21   4.5106e+20   1.1224e+21
power-shifts =    1.9024e+20   1.5859e+21   6.5367e+20   1.3251e+21
power-shifts =    3.8799e+19   1.7373e+21   4.6116e+20   1.1336e+21
power-shifts =    1.9740e+20   1.5787e+21   6.4253e+20   1.3150e+21
power-shifts =    4.2085e+19   1.7340e+21   4.7132e+20   1.1447e+21
power-shifts =    2.0469e+20   1.5714e+21   6.3144e+20   1.3048e+21
power-shifts =    4.5517e+19   1.7306e+21   4.8157e+20   1.1557e+21
power-shifts =    2.1209e+20   1.5640e+21   6.2038e+20   1.2946e+21
power-shifts =    4.9093e+19   1.7270e+21   4.9188e+20   1.1667e+21
power-shifts =    2.1960e+20   1.5565e+21   6.0938e+20   1.2842e+21
power-shifts =    5.2813e+19   1.7233e+21   5.0226e+20   1.1777e+21
power-shifts =    2.2723e+20   1.5489e+21   5.9842e+20   1.2739e+21
power-shifts =    5.6675e+19   1.7194e+21   5.1270e+20   1.1886e+21
power-shifts =    2.3497e+20   1.5411e+21   5.8751e+20   1.2634e+21
power-shifts =    6.0681e+19   1.7154e+21   5.2321e+20   1.1995e+21
power-shifts =    2.4283e+20   1.5333e+21   5.7666e+20   1.2529e+21
power-shifts =    6.4828e+19   1.7113e+21   5.3378e+20   1.2103e+21
power-shifts =    2.5080e+20   1.5253e+21   5.6585e+20   1.2423e+21
power-shifts =    6.9116e+19   1.7070e+21   5.4441e+20   1.2210e+21
power-shifts =    2.5887e+20   1.5172e+21   5.5510e+20   1.2317e+21

   2072  4.166779e+08  3.865156e+08  7.803645e-02  5.704719e-05  0  0  0  5    1  0  0  0  0  0  0  0  0  0
itert =    2072     freqt =    3.0651e-01     wdesf =    1.0000e+00     ishift = -32003
nshftn =     180     nshfts =     240     ndotmn =       4     ndot =     720     ifull = -32003


start of search for mode =       1
fractional accuracy sought =         5.9e-04
w2min,freq2,w2des,w2desf,w2max =
          0.          1.7362e+17   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    4.0673e+18   1.7721e+21   2.6844e+20   8.9578e+20
power-shifts =    7.7355e+19   1.6988e+21   8.8035e+20   1.5077e+21
power-shifts =    4.3366e+18   1.7718e+21   2.7954e+20   9.1121e+20
power-shifts =    8.3631e+19   1.6925e+21   8.6492e+20   1.4966e+21
power-shifts =    4.8751e+18   1.7713e+21   2.9082e+20   9.2663e+20
power-shifts =    9.0151e+19   1.6860e+21   8.4950e+20   1.4853e+21
power-shifts =    5.6826e+18   1.7704e+21   3.0229e+20   9.4203e+20
power-shifts =    9.6915e+19   1.6792e+21   8.3410e+20   1.4738e+21
power-shifts =    6.7586e+18   1.7694e+21   3.1393e+20   9.5743e+20
power-shifts =    1.0392e+20   1.6722e+21   8.1871e+20   1.4622e+21
power-shifts =    8.1036e+18   1.7680e+21   3.2575e+20   9.7280e+20
power-shifts =    1.1116e+20   1.6650e+21   8.0333e+20   1.4504e+21
```

Figure 6.5. First 70 columns of processed output file (continued).

```
power-shifts =    9.7163e+18   1.7664e+21   3.3774e+20   9.8814e+20
power-shifts =    1.1864e+20   1.6575e+21   7.8799e+20   1.4384e+21
power-shifts =    1.1597e+19   1.7645e+21   3.4990e+20   1.0035e+21
power-shifts =    1.2636e+20   1.6498e+21   7.7268e+20   1.4262e+21
power-shifts =    1.3744e+19   1.7624e+21   3.6222e+20   1.0187e+21
power-shifts =    1.3430e+20   1.6418e+21   7.5740e+20   1.4139e+21
power-shifts =    1.6158e+19   1.7600e+21   3.7471e+20   1.0340e+21
power-shifts =    1.4248e+20   1.6337e+21   7.4216e+20   1.4014e+21

power-shifts =    1.8837e+19   1.7573e+21   3.8734e+20   1.0492e+21
power-shifts =    1.5088e+20   1.6252e+21   7.2696e+20   1.3888e+21
power-shifts =    2.1781e+19   1.7544e+21   4.0014e+20   1.0643e+21
power-shifts =    1.5951e+20   1.6166e+21   7.1182e+20   1.3760e+21
power-shifts =    2.4989e+19   1.7511e+21   4.1308e+20   1.0794e+21
power-shifts =    1.6836e+20   1.6078e+21   6.9673e+20   1.3631e+21
power-shifts =    2.8460e+19   1.7477e+21   4.2616e+20   1.0944e+21
power-shifts =    1.7743e+20   1.5987e+21   6.8169e+20   1.3500e+21
power-shifts =    3.2192e+19   1.7439e+21   4.3939e+20   1.1094e+21
power-shifts =    1.8672e+20   1.5894e+21   6.6672e+20   1.3367e+21
power-shifts =    3.6186e+19   1.7399e+21   4.5275e+20   1.1243e+21
power-shifts =    1.9621e+20   1.5799e+21   6.5182e+20   1.3234e+21
power-shifts =    4.0438e+19   1.7357e+21   4.6624e+20   1.1391e+21
power-shifts =    2.0592e+20   1.5702e+21   6.3699e+20   1.3099e+21
power-shifts =    4.4949e+19   1.7312e+21   4.7987e+20   1.1539e+21
power-shifts =    2.1584e+20   1.5603e+21   6.2223e+20   1.2963e+21
power-shifts =    4.9717e+19   1.7264e+21   4.9361e+20   1.1686e+21
power-shifts =    2.2596e+20   1.5502e+21   6.0756e+20   1.2825e+21
power-shifts =    5.4740e+19   1.7214e+21   5.0748e+20   1.1832e+21
power-shifts =    2.3629e+20   1.5398e+21   5.9297e+20   1.2687e+21

power-shifts =    6.0017e+19   1.7161e+21   5.2146e+20   1.1977e+21
power-shifts =    2.4681e+20   1.5293e+21   5.7847e+20   1.2547e+21
power-shifts =    6.5547e+19   1.7106e+21   5.3556e+20   1.2121e+21
power-shifts =    2.5753e+20   1.5186e+21   5.6407e+20   1.2406e+21
power-shifts =    7.1327e+19   1.7046e+21   5.4976e+20   1.2264e+21
power-shifts =
    2792  4.101093e+08  4.098328e+08  6.746629e-04  1.532444e-06  0  0  0  6     1  0  0  0  0  0  0  0  0  0
intermediate result.
iterations, frequency, fractional error:  2792   4.0983e+08       6.7e-04
estimate of next resonance frequency:   6.1795e+08
estimate for fractional contamination in resonance:          1.8e-01
itert =       0    freqt =   2.0834e-01    wdesf =   1.0000e+00    ishift = -32003
nshftn =       4    nshfts =    180    ndotmn =      4     ndot =       8    ifull = -32004


start of search for mode =      2
fractional accuracy sought =        1.0e-03
w2min,freq2,w2des,w2desf,w2max =
         0.              4.0000e+18   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    1.8460e+20   1.7093e+21   6.3117e+20   1.2627e+21
power-shifts =
      8  8.854815e+08  8.854815e+08  1.000000e+00  1.000000e+00  0  0  0  0     0  2  0  0  0  0  0  0  0  0
itert =       8    freqt =   3.7582e-00    wdesf =   1.0000e+00    ishift = -32003
nshftn =     108    nshfts =      4    ndotmn =      4     ndot =     216    ifull = -32004


start of search for mode =      2
fractional accuracy sought =        1.0e-03
w2min,freq2,w2des,w2desf,w2max =
         0.              7.8408e+17   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    4.1870e+18   1.7720e+21   2.7217e+20   9.0095e+20
power-shifts =    8.1569e+19   1.6946e+21   8.7524e+20   1.5040e+21
```

Figure 6.5. First 70 columns of processed output file (continued).

```
power-shifts =    4.9349e+18   1.7713e+21   2.9087e+20   9.2666e+20
power-shifts =    9.2436e+19   1.6838e+21   8.4954e+20   1.4853e+21
power-shifts =    6.4300e+18   1.7698e+21   3.1008e+20   9.5233e+20
power-shifts =    1.0398e+20   1.6722e+21   8.2387e+20   1.4661e+21
power-shifts =    8.6711e+18   1.7675e+21   3.2978e+20   9.7794e+20
power-shifts =    1.1618e+20   1.6600e+21   7.9825e+20   1.4464e+21
power-shifts =    1.1656e+19   1.7645e+21   3.4995e+20   1.0035e+21
power-shifts =    1.2904e+20   1.6472e+21   7.7271e+20   1.4262e+21
power-shifts =    1.5383e+19   1.7608e+21   3.7057e+20   1.0289e+21
power-shifts =    1.4254e+20   1.6337e+21   7.4727e+20   1.4056e+21
power-shifts =    1.9848e+19   1.7563e+21   3.9164e+20   1.0542e+21
power-shifts =    1.5667e+20   1.6195e+21   7.2194e+20   1.3846e+21
power-shifts =    2.5048e+19   1.7511e+21   4.1312e+20   1.0794e+21
power-shifts =    1.7141e+20   1.6048e+21   6.9676e+20   1.3631e+21
power-shifts =    3.0978e+19   1.7452e+21   4.3501e+20   1.1044e+21
power-shifts =    1.8677e+20   1.5894e+21   6.7174e+20   1.3412e+21
power-shifts =    3.7623e+19   1.7386e+21   4.5728e+20   1.1293e+21
power-shifts =    2.0272e+20   1.5735e+21   6.4690e+20   1.3189e+21

power-shifts =    4.5008e+19   1.7312e+21   4.7991e+20   1.1539e+21
power-shifts =    2.1925e+20   1.5569e+21   6.2227e+20   1.2963e+21
power-shifts =    5.3096e+19   1.7231e+21   5.0289e+20   1.1783e+21
power-shifts =    2.3634e+20   1.5399e+21   5.9786e+20   1.2733e+21
power-shifts =    6.1891e+19   1.7143e+21   5.2619e+20   1.2025e+21
power-shifts =    2.5399e+20   1.5222e+21   5.7370e+20   1.2500e+21
power-shifts =    7.1384e+19   1.7048e+21   5.4980e+20   1.2264e+21
power-shifts =
     224  7.350283e+08  7.294980e+08  7.580928e-03  6.881263e-05  0  0  0  0     0  2  0  0  0  0  0  0  0  0
itert =      224     freqt =   4.4274e-01     wdesf =    1.0000e+00     ishift = -32003
nshftn =     120      nshfts =     108     ndotmn =      4     ndot =     480     ifull = -32003
itert =      224     freqt =   4.4274e-01     wdesf =    1.0000e+00     ishift = -32003
nshftn =     120      nshfts =     108     ndotmn =      4     ndot =     480     ifull = -32003
itert =      224     freqt =   4.4274e-01     wdesf =    1.0000e+00     ishift = -32003
nshftn =     120      nshfts =     108     ndotmn =      4     ndot =     480     ifull = -32003


start of search for mode =      2
fractional accuracy sought =        1.0e-03
w2min,freq2,w2des,w2desf,w2max =
            0.            5.4034e+17   4.0000e+18   4.0000e+18   1.7721e+21
power-shifts =    4.1515e+18   1.7720e+21   2.7123e+20   8.9966e+20
power-shifts =    7.5865e+19   1.7003e+21   8.7652e+20   1.5049e+21
power-shifts =    4.7573e+18   1.7714e+21   2.8802e+20   9.2279e+20
power-shifts =    8.5278e+19   1.6909e+21   8.5338e+20   1.4882e+21
power-shifts =    5.9685e+18   1.7702e+21   3.0522e+20   9.4590e+20
power-shifts =    9.5241e+19   1.6809e+21   8.3027e+20   1.4710e+21
power-shifts =    7.7842e+18   1.7684e+21   3.2282e+20   9.6898e+20
power-shifts =    1.0575e+20   1.6704e+21   8.0720e+20   1.4534e+21
power-shifts =    1.0203e+19   1.7660e+21   3.4080e+20   9.9199e+20
power-shifts =    1.1679e+20   1.6594e+21   7.8418e+20   1.4354e+21
power-shifts =    1.3224e+19   1.7629e+21   3.5916e+20   1.0149e+21
power-shifts =    1.2836e+20   1.6478e+21   7.6124e+20   1.4170e+21
power-shifts =    1.6844e+19   1.7593e+21   3.7788e+20   1.0378e+21
power-shifts =    1.4045e+20   1.6357e+21   7.3838e+20   1.3983e+21
power-shifts =    2.1062e+19   1.7551e+21   3.9696e+20   1.0605e+21
power-shifts =    1.5306e+20   1.6231e+21   7.1562e+20   1.3792e+21
power-shifts =    2.5873e+19   1.7503e+21   4.1637e+20   1.0832e+21
power-shifts =    1.6617e+20   1.6100e+21   6.9299e+20   1.3598e+21
power-shifts =    3.1276e+19   1.7449e+21   4.3610e+20   1.1057e+21
power-shifts =    1.7977e+20   1.5964e+21   6.7048e+20   1.3401e+21

power-shifts =    3.7266e+19   1.7389e+21   4.5614e+20   1.1280e+21
```

Figure 6.5. First 70 columns of processed output file (continued).

```
 power-shifts =    1.9386e+20   1.5823e+21   6.4813e+20   1.3200e+21
 power-shifts =    4.3839e+19   1.7323e+21   4.7648e+20   1.1502e+21
 power-shifts =    2.0842e+20   1.5678e+21   6.2594e+20   1.2997e+21
 power-shifts =    5.0990e+19   1.7252e+21   4.9710e+20   1.1722e+21
 power-shifts =    2.2345e+20   1.5527e+21   6.0393e+20   1.2791e+21
 power-shifts =    5.8715e+19   1.7175e+21   5.1799e+20   1.1941e+21
 power-shifts =    2.3894e+20   1.5372e+21   5.8212e+20   1.2582e+21
 power-shifts =    6.7009e+19   1.7092e+21   5.3913e+20   1.2157e+21
 power-shifts =    2.5487e+20   1.5213e+21   5.6051e+20   1.2370e+21
 power-shifts =
     704  7.250168e+08  7.350001e+08  2.276690e-05  2.617262e-08  0  0  0  0     1  3  0  0  0  0  0  0  0  0
 intermediate result.
 iterations, frequency, fractional error:   704   7.3500e+08       2.3e-05
 estimate of next resonance frequency:   8.2241e+08
 estimate for fractional contamination in resonance:       1.2e-01
1summary of simulation parameters
0electromagnetic parameters
          number of cells      =    269001
          number of time steps =         2
          product              =    538002
0model parameters
          number of structures =        12
0output parameters

 final frequency results.
 resonance #, frequency, and accuracy:   1   4.0983e+08   2.7650e+05
 resonance #, frequency, and accuracy:   2   7.3500e+08   1.6734e+04
 total number of iterations:       3496
 total number of deflations:         10
 total number of renormalizations:         0
 program sos - simulation completed.
```

Figure 6.5. **First 70 columns of processed output file (concluded).**

frequency was 4.1005E+08 with an error of 1.6167E+5. The second resonant frequency was with an error of 1.9969E+06.

6.2 CO-PLANAR TRANSMISSION LINE

The second example is a time-domain run of a simple co-planar transmission line. A 'viewplot' of a section of a coplanar transmission line, as generated by SOS, is shown in Figure 6.6. The input deck is listed in Figure 6.7. This transmission line has one thin strip surrounded by two co-planar strips, all lying on a dielectric. Below the dielectric is a ground plane. In this example, we will demonstrate the use of the VOLTAGE, DIELECTRIC, SYMMETRY and other related commands. The goal of this simulation is to measure the effective resistance of the strip as a wave passes through it. This is done by integrating the electric field to calculate the voltage and integrating the magnetic field to find the current. Various plots of the electric and magnetic fields are given in the following figures.

6.2.1 Parameter Definition

As in the previous example, several variables that are used in later commands are defined at the beginning of the input deck. Most of these are used in the same way as in 6.1.3. The DEFINE command can be used as a calculator to calculate one parameter based on the values of others. In this example, we calculate the number of time steps required for the run. We want this to be such that the pulse has time to travel the entire length of the x3 axis. The distance of the line is calculated and stored in the variable DISTANCE,

```
DEFINE "DISTANCE = NGRDZ*ZCELLSZ" ;
```

The velocity of light in the dielectric is c divided by the square root of epsilon,

```
DEFINE "VELOCITY = 3E8 / ROOTEPS ;
```

The amount of time it takes the pulse to travel the line is the distance divided by the speed of light in the dielectric. The total time needed for the simulation is this time plus the period of the pulse. Once this time is found, the number of time steps is the total time divided by the time step size. In this way, the size of the line can be changed, and the number of time steps will be

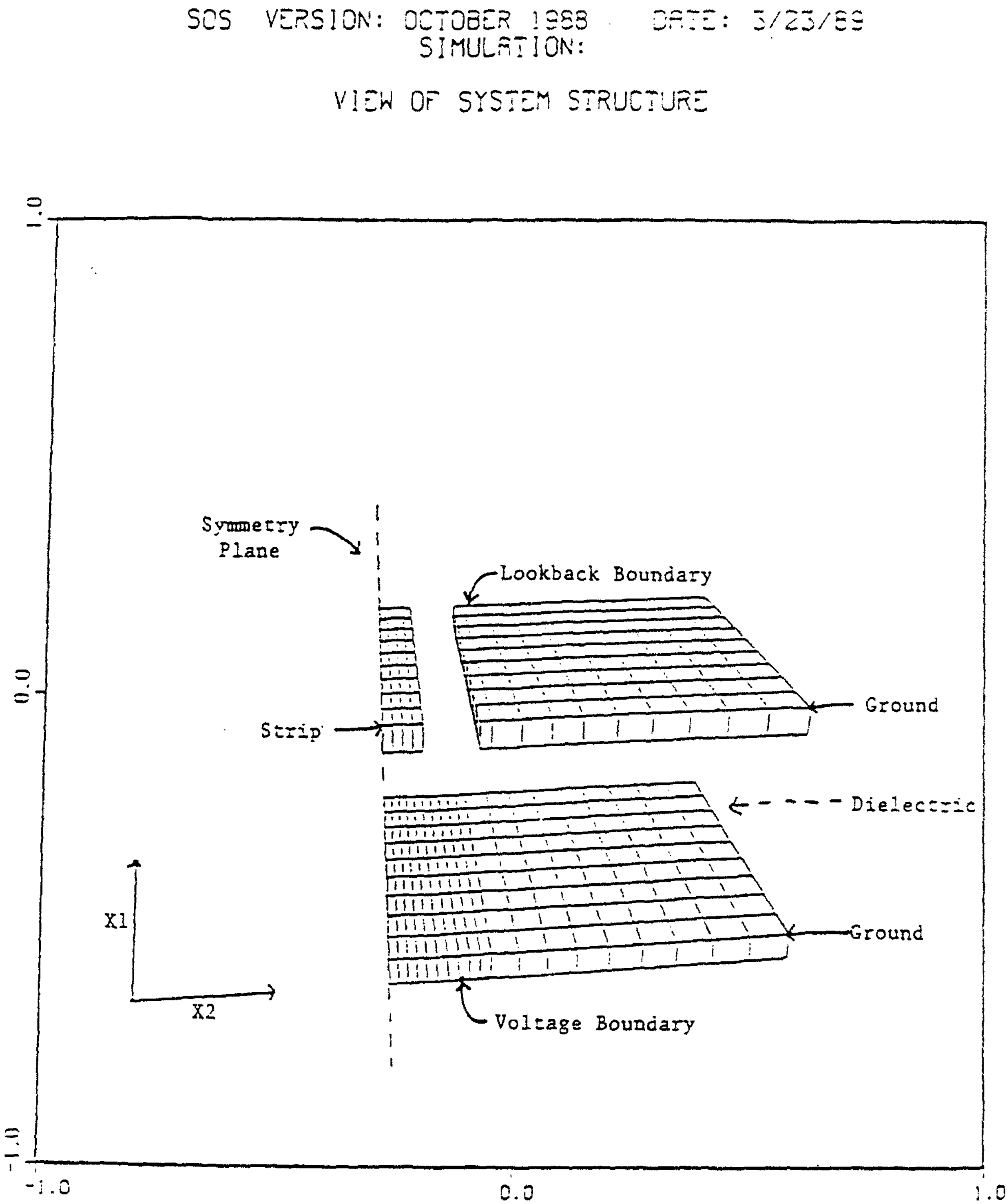


Figure 6.6. Perspective plot of co-planar transmission line.

```
TITLE "Co-planar Transmission Line" ;
COMMENT "This simulation tests the transmission vs. reflection
         of a simple micro-strip form MMIC device.  The purpose
         is to test the ability of SOS to simulate MMIC devices." ;

COMMENT "Set up variables for use in command statements" ;
DEFINE "IXS = 21" ;
DEFINE "IXF = 22" ;
DEFINE "ISTRIPW = 5" ;
DEFINE "ISIDEW = 10" ;
DEFINE "NPX = 40" ;
DEFINE "NPY = 20" ;
DEFINE "NPZ = 10" ;

DEFINE "NGRDX = NPX + 1" ;
DEFINE "NGRDY = NPY + 1" ;
DEFINE "NGRDZ = NPZ + 1" ;
DEFINE "ZCELLSZ = 60.E-6" ;

COMMENT "Define parameters for time domain run.  The total
         time is calculated to be the time it takes the
         pulse to travel the length of the guide." ;
DEFINE "TSTEP = 3.E-14" ;
DEFINE "EPSILON = 12.5" ;
DEFINE "DISTANCE = NGRDZ * ZCELLSZ" ;
DEFINE "ROOTEPS = EPSILON ** .5" ;
DEFINE "VELOCITY = 3E8 / ROOTEPS" ;
DEFINE "TIME = DISTANCE / VELOCITY" ;
DEFINE "TIME = TIME + 8E-12" ;
DEFINE "NSTEP = TIME / TSTEP" ;

COMMENT "Set up grid" ;
DEFINE "NXLEFT = NPX - 20 - 1" ;
X1GRID FUNCTION NGRDX 1 0.0
       20  27.E-6  400.E-6
        1  13.E-6   13.E-6
   NXLEFT  13.E-6 1200.E-6 ;
X2GRID FUNCTION NGRDY 1 0.0
        4  13.E-6   54.E-6
        5  15.E-6   76.E-6
        1  16.E-6   16.E-6
       10  40.E-6  500.E-6 ;
X3GRID UNIFORM NGRDZ 1 0.0 ZCELLSZ ;

FIELDS CENTERED NSTEP TSTEP ;
COURANT SEARCH ;
```

Figure 6.7. Input deck.

```
COMMENT "Set up geometry" ;
STRUCTURE CONFORMAL BOTTOM SOLID 0 1         1 NGRDY       1 NGRDZ ;
STRUCTURE CONFORMAL TOP    SOLID NPX NGRDX  1 NGRDY       1 NGRDZ ;
STRUCTURE CONFORMAL SIDE   SOLID IXS IXF    ISIDEW NGRDY  1 NGRDZ ;
STRUCTURE CONFORMAL STRIP  SOLID IXS IXF    1 ISTRIPW     1 NGRDZ ;

COMMENT "Set up mirrors at bounds of Y" ;
SYMMETRY MIRROR ALIGN       1 NGRDX  1 1          1 NGRDZ ;
SYMMETRY MIRROR ANTI-ALIGN  1 NGRDX  NGRDY NGRDY  1 NGRDZ ;

COMMENT "Dielectric corresponding to GaAs below microstrip" ;
DIELECTRIC 0 NULL EPSILON 0 0  1 IXS    1 NGRDY       1 NGRDZ ;
INTERFACE EPSILON 1. 1         IXS IXS  ISIDEW NGRDY  1 NGRDZ ;

COMMENT "LOOKBACK boundary on end of Z" ;
LOOKBACK 1.0 1.0    ANTI-ALIGN  IXF NGRDX  1 NGRDY       NGRDZ NGRDZ ;
LOOKBACK 1.0  .2829 ANTI-ALIGN  1 IXS      1 NGRDY       NGRDZ NGRDZ ;
LOOKBACK 1.0  .385  ANTI-ALIGN  IXS IXF    ISIDEW NGRDY  NGRDZ NGRDZ ;

COMMENT "VOLTAGE boundary on start of Z" ;
POISSON 4
   SIDE   0.
   BOTTOM 0.
   TOP     .1
   STRIP  4.
   X1 0. 1.  400  1 NGRDX  1 NGRDY  1 1 ;
FUNCTION INJECT DATA 13
0.00     .0
5.20E-13 .041122615
1.30E-12 .1722997
1.92E-12 .4686041
2.85E-12 .8095462
3.43E-12 .9507271
3.95E-12 .9996145
4.47E-12 .9663200
5.03E-12 .8451268
5.80E-12 .5792189
6.82E-12 .1997917
7.41E-12 .052728862
8.00E-12 .0 ;
DEFINE "SCALE = 4" ;
VOLTAGE FIELD INJECT 0. SCALE 1.0    ALIGN  IXF NGRDX  1 NGRDY       1
1 ;
VOLTAGE FIELD INJECT 0. SCALE  .2829 ALIGN  1 IXS      1 NGRDY       1
1 ;
VOLTAGE FIELD INJECT 0. SCALE  .385  ALIGN  IXS IXF    ISIDEW NGRDY  1
1 ;
```

Figure 6.7. Input deck (continued).

```
GRAPHICS SYSTEM ;
C SHOW GRID 1 IXS ;
C SHOW MODEL 1 IXS ;
C SHOW MODEL 3 2 ;
C SHOW GRID 3 2 ;

COMMENT "Integrate between ground and strip to get potential" ;
OBSERVE 10 1    FIELD 4  0 1.2E-10  0 25.E9  E1  1 IXS  2 2  2 2 ;
OBSERVE 10 1    FIELD 4  0 1.2E-10  0 25.E9  E1  1 IXS  2 2  NPZ NPZ ;
COMMENT "Integrate around small strip to calculate current" ;
OBSERVE 10 1    FIELD 4  0 1.2E-10  0 25.E9  B2  13 13  1 7  2 2 ;
OBSERVE 10 1    FIELD 4  0 1.2E-10  0 25.E9  B1  13  8  7 7  2 2 ;
OBSERVE 10 1    FIELD 4  0 1.2E-10  0 25.E9  B2   8  8  7 1  2 2 ;

TIMER RTIM PERIODIC 80 99999 80 ;
RANGE RTIM  1 0 0  E1  5 5  10 10  1 10 ;

TIMER VTIM DISCRETE 250 ;
VECTOR VTIM E X3 X1  0 1.6E-03  2.6E-6 2.6E-6  0 DISTANCE   40 40 ;
VECTOR VTIM B X2 X1  0 1.6E-03  0 6.25E-4      .1E-3 .1E-3  40 40 ;

START ;
STOP ;
```

Figure 6.7. Input deck (concluded).

recalculated automatically. The size of the time step is fixed by the Courant stability condition (see COURANT.)

6.2.2 Grid Definition

The grid is defined so that there are 400 micrometers between the ground and the co-planar strips. These strips are 13 micrometers thick. The FUNCTION option is used in the X1 and X2 directions, allowing for smoothly varying grid sizes. The grid size on the X1 axis gets much larger farther from the strip, because this region does not need to be modeled as closely. The X3 axis is defined with a uniform grid.

6.2.3 Symmetry Mirrors

The simulation is symmetric about the X2 = 0 point. In order to have increased resolution, we only model one half of the simulation and use a symmetry mirror for the other half. This is done by placing a mirror on the X2 = 0 plane. This command is,

SYMMETRY MIRROR ALIGN 1 NGRDX 1 1 1 NGRDZ ;

The surface normal asnor = ALIGN indicates that the geometry on the positive side of the mirror is to be reflected. A mirror with asnor = anti-align is placed at the other end of the Y axis.

6.2.4 Dielectric and Interface Commands

In this simulation we want a dielectric with a relative dielectric constant of 12.5 to be between the ground plate and the co-planar strips. The command that does this is

DIELECTRIC 0 NULL EPSILON 0 0 1 IXS 1 NGRDY 1 NGRDZ ;

This dielectric has a relative dielectric constant of EPSILON which is defined earlier as 12.5. Photo-induced currents are not relevant in the simulation, so there is no photon source name. The prompt conductivity constant and the Compton current constant are both zero. The dielectric fills the block between the cells X1 = 1 and X1 = IXS which is the height of the strips and is defined to be 10. The INTERFACE command is used to specify the boundary interfaces of the dielectric. In this example there is only one interface on the X1 = IXS plane. The command is

INTERFACE EPSILON 1.0 1 IXS IXS 151 DEW NGRDY 1 NGRDZ ;

The dielectric constant on the negative side of the interface is EPSILON (12.5), and on the positive side it is 1.0 (air). The axis normal to the interface is the X1 axis.

6.2.5 Voltage and Lookback Commands

Next in the input deck, we want to set up the time-dependent boundary conditions for the first and last X3 planes. We want a pulse to come in the line from the X3 = 1 plane and leave through the NGRDZ plane. The incoming pulse is created using a VOLTAGE command, while the transmitted pulse is done with a LOOKBACK command. One of the commands that specifies the TEM electromagnetic wave entering the simulation is

```
VOLTAGE FIELD INJECT 0. SCALE  1.0  ALIGN IXF NGRDX 1
        NGRDY 1 1 ;
```

The name of the function that specifies the input wave is INJECT. The values of this pulse are given in the FUNCTION INJECT command later in the input deck. There is no time delay in the run. The scale is a dimensionless factor that scales the pulse. In this example the SCALE is defined to be 4 to produce a 4-volt pulse using the Poisson solution with 1 volt. The area specified by the above command is the part of the X3 = 1 plane above the two strips. This region is filled with air so the phase velocity factor (v/c) is 1. In the commands that follow, the phase velocity factors are .385 and .2829, one being in the conductor and the other in the dielectric, respectively. Note that these are all less than or equal to one. Also, the order of the VOLTAGE commands is important

The LOOKBACK command provides an outlet for an electromagnetic wave to escape from the simulation. This changes the simulation from a cavity to a transmission line. The LOOKBACK command's syntax is very similar to the VOLTAGE command except that no driving function is required. There are three LOOKBACK commands in the input deck, one for each type of material in the X3 = NGRDZ plane. The command for the top part of the simulation is

```
LOOKBACK 1.0 ANTI IXF NGRDX 1 NGRDY  NGRDZ NGRDZ ;
```

The relative phase velocity in air is 1.0. The surface normal alignment is anti-align, because the simulation is on the inside of the X3 = NGRDZ plane. The other two commands for the dielectric and conductor are similar, with different phase velocity factors.

Note that for the VOLTAGE and LOOKBACK commands, only one grid plane is used for each. Many different commands for each can be used, but they must all be in the same plane.

The POISSON command specifies an electrostatic solution at the boundary surfaces that is to be added to the incoming pulse. In the POISSON command the conductors of the simulation must be specified as well as their initial surface potentials. In this run, the side and bottom are grounded (V=0). The strip is at 4 volts and the top is at 0.1 volts. A conductor is placed at the top to provide some boundary there. The Poisson solution is to be calculated for the X3=1 plane. The command is

```
POISSON 4 SIDE 0. BOTTOM 0. TOP .1 STRIP 4.  X1 0. 1. 400 1
        NGRDX  1 NGRDY 1 1 ;
```

The coordinate with the largest gradient is the X1 direction. The linear coefficient and the over-relaxation coefficient are both 1.0. The command specifies that 400 iterations are to be taken.

6.2.6 Graphical Output

There are several ways that we can watch the pulse travel across the line and see how it is distorted. The OBSERVE command plots, as a function of time, a specific line integral of the electric field, magnetic field, or current. The RANGE command plots the values of one field along a line at one time. Several of these can be made to watch the pulse travel down the line. A vector plot can be made of the fields at a given time in a plane. The first thing we want to do is compare the input voltage pulse with the output pulse to see if there is any distortion in the signal. To do this we use the OBSERVE command and integrate the electric field from the ground to the center strip, giving us the potential difference as a function of time. We do this at the beginning of the line and at the end. The command for the voltage at the beginning is

```
OBSERVE 10 1 FIELD 4 0 1.2E-10 0 25.E9  E1  1 IXS  2 2  2 2 ;
```

The X1 component of the electric field is to be integrated from (1,2,2) to (IXS,2,2) every 10 time steps. No Fourier transform is required. The results of this command and the one at the end of the line are given in Figures 6.8 and 6.9. The shape of the pulse is distorted only slightly. The time delay for the time required by the signal to travel down the line can be seen by comparing these two figures.

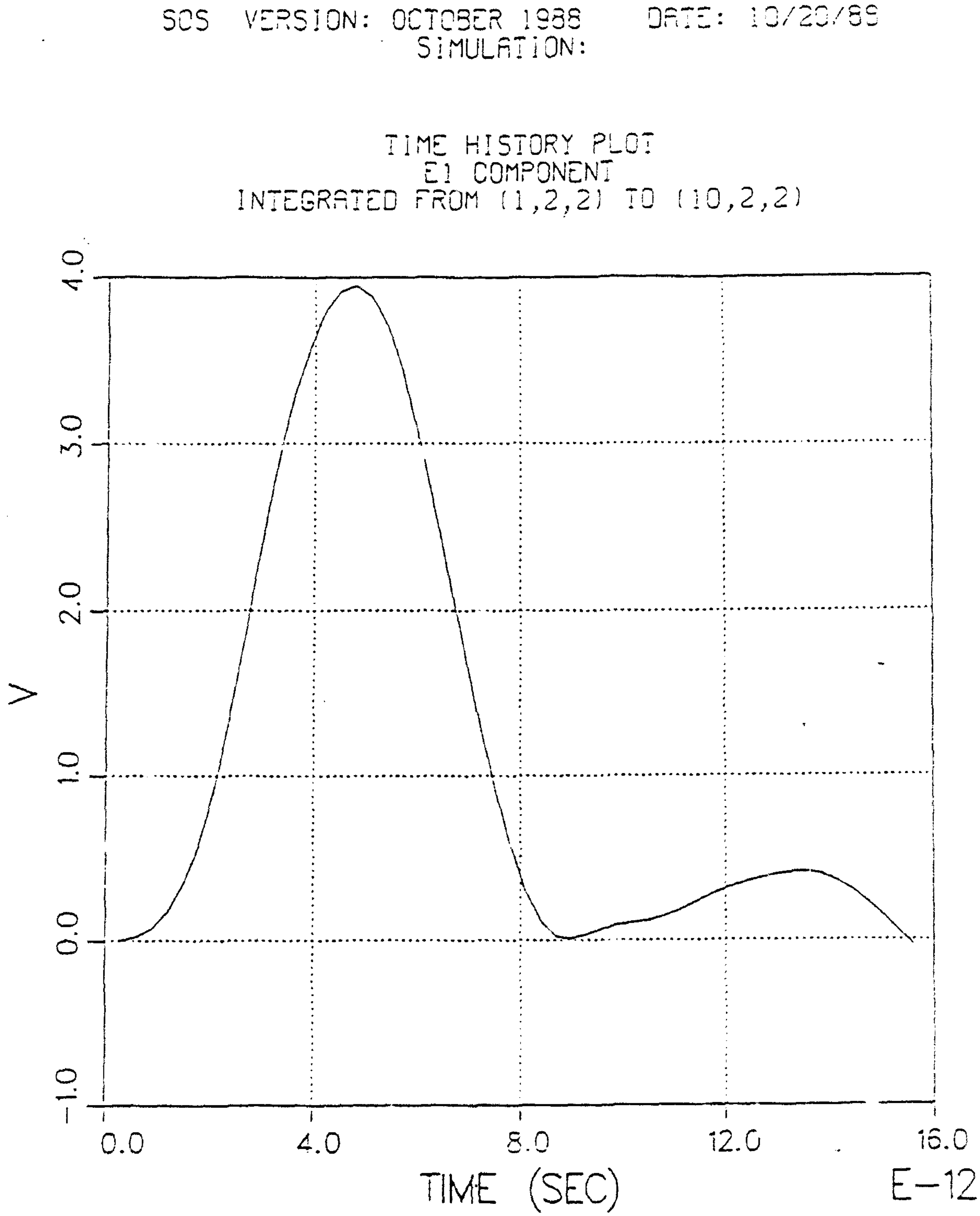


Figure 6.8. Incoming pulse.

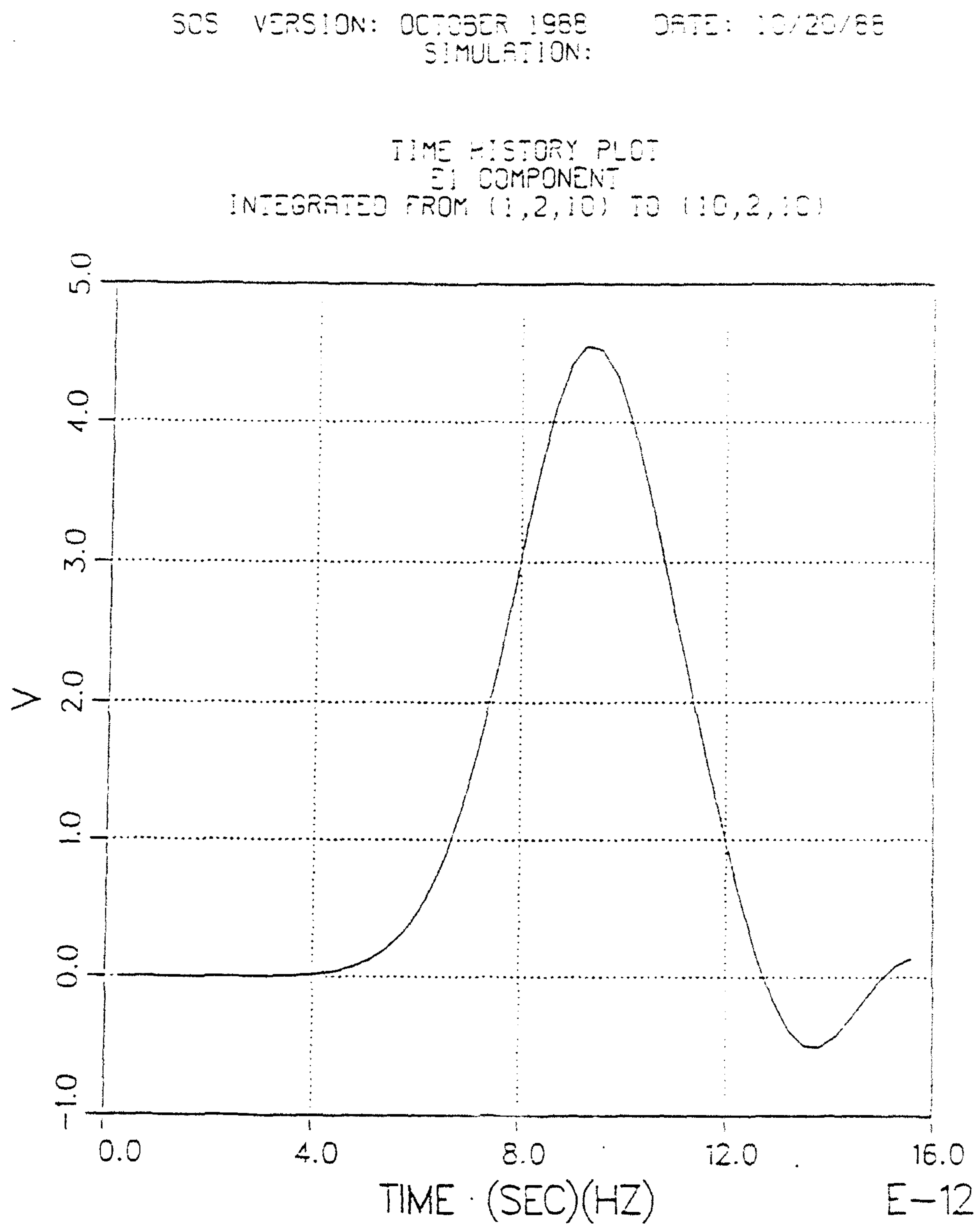


Figure 6.9. Transmitted pulse.

The OBSERVE command can be used to calculate the current down the line, and from this we can calculate the effective resistance of the transmission line. The current can be calculated by integrating the magnetic field around the center strip. From Ampere's law this is equal to the current through at any given time. The commands for carrying out this integration are in the example input deck in Figure 6.7. The integrals were carried out, and, using Ohm's law, the resistance was calculated to be about 62 ohms. The correct resistance is near 50 ohms. A more accurate solution can be found if the top plate is moved farther away.

The RANGE plot taken as the pulse is halfway down the line as shown in Figure 6.10. The command that generates this and many other range plots at different times is

```
RANGE RTIM 1 0 0 E1 5 5 10 10 1 10 ;
```

A range plot is made every 80 time steps. The X1 component of the electric field along the line (5,10,1) to (5,10,10) is plotted. By looking at all the range plots, one can see the pulse travel down the line. The last two plots are generated by two VECTOR commands. The electric and magnetic fields for different planes are plotted at the middle time step. The commands used to plot the electric field are

```
TIMER VTIM DISCRETE 250 ;
VECTOR E X3 X1  0 1.6E-3 2.6E-6
2.6E-6 0 DISTANCE 40 40 ;
```

These commands generates one plot at the 250th time step. The X3 and X1 components of the electric field are to be plotted against each other in the X2=2.6E-6 plane. The last two arguments specify that there are to be 40 vectors in each direction. This plot is given in Figure 6.11. The conductor is visible (the center strip where the E-field is zero), as is the wave. One interesting bit of physics this plot shows is the difference in the velocity of the wave above and below the coplanar line. There is a dielectric below the coplanar line, so the speed of light in this region is smaller than in air above.

The second vector plot, shown in Figure 6.12, is of the magnetic field in the plane X3=1.0E-4. The two strips are visible as is the curl of the magnetic field, indicating a current down the center strip. This plot shows the magnetic field in the region in which it is integrated by the OBSERVE command.

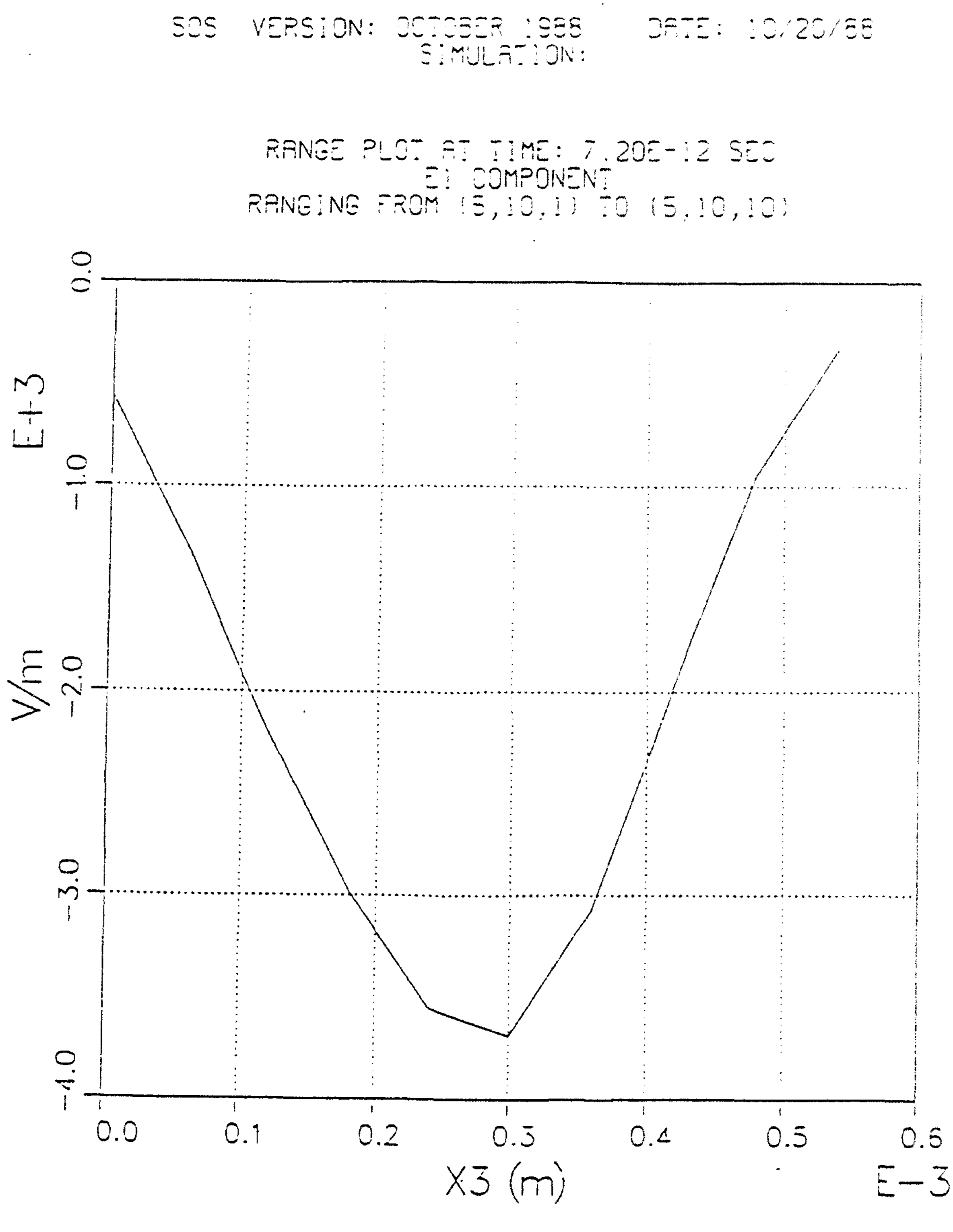


Figure 6.10. **View of pulse propagating down the line.**

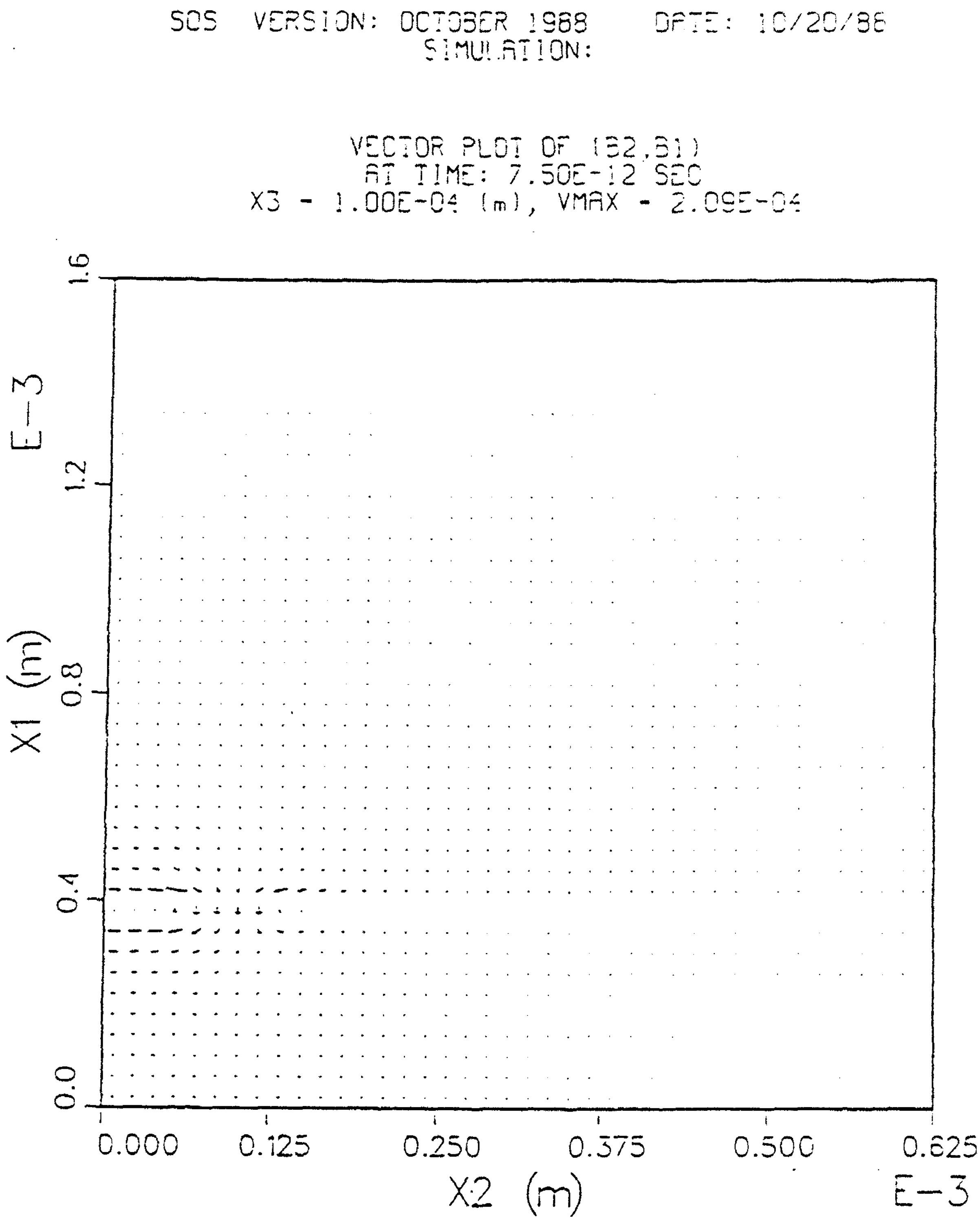


Figure 6.11. Vector plot of electric fields.

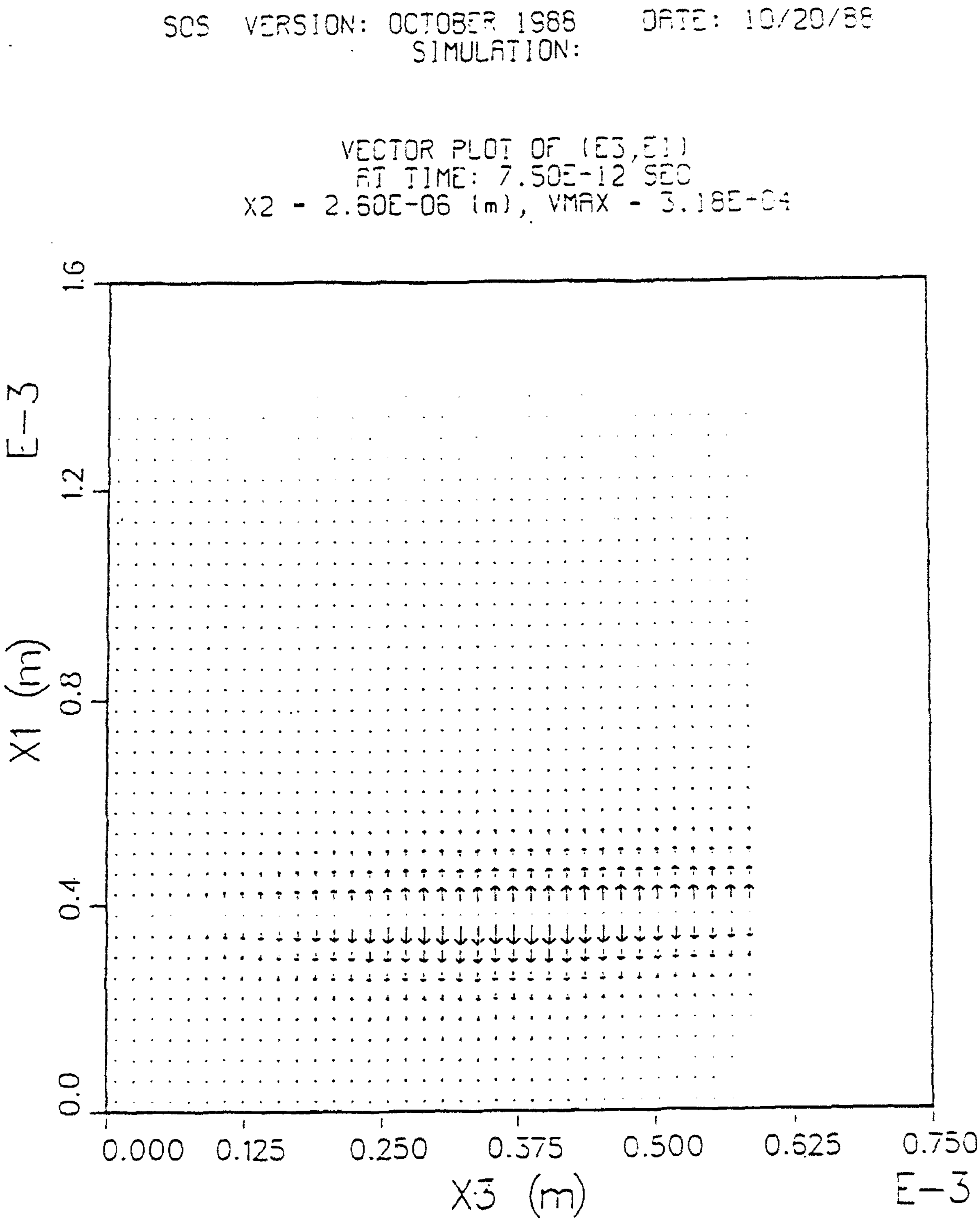


Figure 6.12. **Vector plot of magnetic fields.**

APPENDIX A

COMMAND FUNCTION SUMMARY (BY GROUP)

INPUT

C	-	Puts a comment into input that does not echo on output.
CALL	-	Reads and processes a command file.
COMMENT	-	Places a comment into both input and output files.
CONVERT	-	Assigns a length or angle in user-specified units to a variable.
DEFINE	-	Defines variables for use in other commands.
DO	-	Enables loop over commands in the input file.
ECHO	-	Turns on echo of processed commands to output file.
ELSE	-	Initiates conditional or unconditional branch execution.
ENDDO	-	Enables loop over commands in the input file.
ENDIF	-	Terminates conditional execution logic.
FUNCTION	-	Defines functions to be referenced by other commands.
IF	-	Provides conditional execution of commands.
NOECHO	-	Turns off echo of processed commands to output file.
RETURN	-	Returns from a call to a command file.
Z	-	Preserves a command in the input file without executing it.

CONTROL

JOURNAL	-	Controls journal facility.
PAUSE	-	Pauses on errors of various severities.
RECORD	-	Records the simulation to disk for later continuation.
RESTART	-	Reads in a previously recorded simulation for continuation.
START	-	Interrupts the processing of input and initiates the simulation.
STOP	-	Terminates execution unconditionally.
SUPPRESS	-	Controls output of processed commands in the output file.
TERMINATE	-	To terminate on errors of various severities.
TIMEOUT	-	Sets aside CPU time for final output.

GEOMETRY

CABLE	-	Specifies coaxial cable drive for electromagnetic excitation of simulation space.
CAPACITOR	-	Specifies capacitive gap location and properties.
DAMPER	-	Specifies membrane damper location and properties.
DIELECTRIC	-	Specifies grid-conformal dielectric blocks.
DIPOLE	-	Specifies a dipole drive surface area.
ENGRID	-	Converts coordinate to grid index.
FREESPACE	-	Absorbs outgoing waves as they leave the simulation.
INTERFACE	-	Specifies boundary interfaces between regions of dielectric.
LOOKBACK	-	Absorbs outgoing waves as they leave the simulation.

PHOTON	-	Specifies photon sources.
SNAPGRID	-	Snaps coordinate to the spatial grid.
STRUCTURE	-	Specifies conformal and non-conformal perfect conductors.
STRUT	-	Specifies strut sub-grid model.
SYMMETRY	-	Specifies symmetry boundaries.
SYSTEM	-	Specifies the coordinate system.
UNGRID	-	Converts grid index into real coordinate.
VOLTAGE	-	Specifies a time-dependent voltage to be applied at a simulation boundary.
X1GRID	-	Generates a spatial grid in the x_1-coordinate.
X2GRID	-	Generates a spatial grid in the x_2-coordinate.
X3GRID	-	Generates a spatial grid in the x_3-coordinate.

<u>FIELDS</u>

CLIGHT	-	Alters the free-space permittivity and permeability.
COURANT	-	Performs a stability check on the electromagnetic field algorithm.
FIELDS	-	Specifies electromagnetic field algorithm and parameters.
FREQUENCY	-	Specifies algorithm to find resonant frequencies and fields.
POISSON	-	Specifies an electrostatic solution at a boundary surface.
PRESCRIBE	-	Prescribes analytical fields for particle kinematics.
PRESET	-	Presets electric and magnetic fields.

PARTICLES

BEXTERNAL	- Specifies spatially independent static magnetic fields.
BEX-READ	- Specifies spatially varying static magnetic fields.
CHEMISTRY	- Specifies treatment of ionized air.
CURRENTS	- Selects options in the current density algorithm.
EMISSION	- Specifies parameters and surfaces for field, photo, and beam emission.
KINEMATICS	- Specifies particle kinematics algorithm.
PHASESPACE	- Plots particle phase space.
STATISTICS	- Specifies times at which particle statistics are collected and printed.
TAGGING	- Specifies which particles are plotted on output graphics.
TRAJECTORY	- Specifies graphical output of particle trajectories.

OUTPUT

BALANCE	- Specifies frequency and range of electromagnetic and particle kinetic energy calculations.
DATAFILE	- Specifies diagnostic output data files to be written for storage and post-processing.
DIAGNOSE	- Specifies diagnostic output.
DISPLAY	- Displays spatial grid and structure.
DUMP	- Specifies output for post-processing.
GRAPHICS	- Specifies output plots on system graphics or line-printer graphics.

LINPRINT	- Specifies printing a two-dimensional map of field components in scientific notation.
LOGPRINT	- Specifies printing two-dimensional maps of field components in single-integer, powers- of-ten notation.
MESSAGE	- Controls error message printing.
OBSERVE	- Specifies simulation variables to be plotted as a function of time or frequency.
RANGE	- Specifies simulation variables to be plotted as functions of a linear coordinate.
TITLE	- Specifies a unique identification for the simulation.
VECTOR	- Specifies vector plots of two-dimensional fields.
VIEW	- Specifies three-dimensional perspective of system structure.

This page is intentionally left blank.